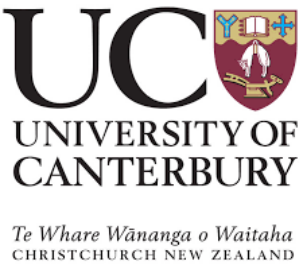


Theoretical and Experimental Aerodynamic Thrust Studies of Propellers at Angles of Incidence

by

Rafael Levy Rubin

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

College of Engineering
Department of Mechanical Engineering
University of Canterbury
Christchurch, New Zealand

August 2021

Abstract

The increasing utilization of small electric unmanned vehicles, in recent years, with vertical take off and landing capabilities, is leading to project demands that include the requirement for propellers to operate at a wide range of flight angles, in many cases. Understanding of the thrust performance in this operational environment is essential. Often, low cost, off-the shelf propellers are used as a project choice, for which a non-complex straightforward thrust performance model is desired. This research focuses on the understanding of propellers thrust performance at angles of incidence AoA, with emphasis on the classical momentum theory, or Actuator Disk Model. The Actuator Disk Model thrust formula was mathematically expanded in series, and divided in two parts to show that propellers at incidence comprise an axial and a wing lift equivalent component. Both components share a common induced speed w. This is done by considering an enhanced disk area for momentum balance, to match Glauert's Hypothesis mass flowrate. To shed light on the theoretical developments, a series of wind tunnel tests were conducted on a few different small-scale fixed-pitch propellers, with blade pitch angles up to 23^o, and advance ratio J values up to 1.2, at angles of incidence ranging from 0° to 90°. The wing component is shown to grow with AoA, being generally negligible at low angles. The slope of this growth, or the sensitivity to AoA, also increases with the airflow velocity V. The axial component decreases with V, for all angles. The generally observed thrust increase with AoA is explained by the theory, to be mostly due to the wing component contribution. The theory also explains why at around $AoA \approx 60^o$ and higher, propellers inherently behave differently than at lower angles. While thrust decreases with V at lower angles, it grows with airspeed, past a certain angle, in most cases around or higher than 60^o. This behavioral inversion happens as the wing component positive sensitivity to V overcomes the negative sensitivity of the axial component. Different propellers will have different angles of thrust behavioural inversion AoA_{inv}, past which, and under increasing velocity, the propeller can be considered to behave as a wing. A simplified formula is presented for predicting thrust at a given angle, based only on propellers data at $AoA = 0^o$, regardless of blade geometry. The simplified formula offers reasonable results up to J values, not approaching the windmill state.

In loving memory of my dear father Fred Alexander Rubin

Acknowledgements

God, Almighty, above all.

I wish to express my gratitude to the people who helped me accomplished this endeavor, in so many different ways.

First to my family, for being by my side, daring to leave everything behind, and to look forward with faith, hope, patience, commitment, and love, accompanying me in this uncertain, but exciting adventure called life.

Special thanks and admiration to my supervisor Prof. Dan Zhao at University of Canterbury, whose pragmatism, competence, experience, and guidance led me to the right path in this research.

Many thanks to my co-supervisor Prof. Edison Gonçalves of the Department of Mechanical Engineering at University of São Paulo - USP, Brazil, whose friendship, and good advise helped me continue, at times of uncertainty.

Thanks to my examiners for taking the time to review the thesis, for their comments, and suggestions.

To all the friends and colleagues who shared moments of fun, good talk, or helped me with Matlab, Solidworks, Latex, 3D-printing, composite manufacturing, with labs work, or whatever else. In alphabetical order: Angus McGregor, Arun Manickavasagam, Benjamin Murton, Daniel Bishop, Daniel Morris, Emeric Jago, Feng Cao (Frank), Jack Davies, James Ramsay, Mathew Furkert, Michael Coe, Ming Bai, Patricio Gallardo, Siliang Ni, Sina Ghafoorpoor, Tao Cai, Teun Brookhuis, and to all the guys at UC Aerospace who helped with the hobby rocket launch. Lastly, to the technicians and engineers who helped me accomplish a lot of lab tests, or with any other issues: Adam Latham, Bill Mohr, Bruce Robertson, David Fanner, David Read, Gary Cotton, Julian Murphy, Julian Phillips, Kenneth Brown, Natalia Kabaliuk, Paul Southward, Richard Jehan, and Tony Doyle. Thanks also to the guys outside UC, who helped with prototypes: Graham Tully (GT composites), and Stephen Smith (Smith CNC patterns).

Publications

This thesis is an original work by Rafael L. Rubin. It includes material reported in the following publications:

R. L. Rubin and D. Zhao, "New Development of Classical Actuator Disk Model for Propellers at Incidence", *AIAA Journal*, Vol. 59, No. 3, 2021, pp. 1040–1059. DOI:10.2514/1.J059734. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.J059734

R. L. Rubin and D. Zhao, "Wind Tunnel Investigation of Propellers at Incidence", presented at the *1st International Conference for Thermal Fluid Dynamics and Control.* (2-6 August 2019), Doubletree Hilton Hotel, Christchurch, NZ.

Contents

Nomenclature

AoA	propeller's angle of incidence, deg
AoA_{inv}	angle of thrust behavioural inversion, deg
C_T	thrust coefficient, $= T/\rho n^2 D^4$
C_Q	torque coefficient, $= Q/\rho n^2 D^5$
D	propeller's diameter, m or in
e	theoretical entrainment factor
$\widehat{\mathbf{e}}_{\mathbf{T}}$	direction versor of thrust force, or propeller axial direction
$\widehat{\mathbf{e}}_{\mathbf{N}}$	direction versor of propeller normal force, due to angle of incidence
F_x	force measured by wind tunnel balance on x axis, net of tare load, N
F_y	force measured by wind tunnel balance on y axis, net of tare load, N
F_z	force measured by wind tunnel balance on z axis, net of tare load, N
J	advance ratio, $= V/nD$
$\dot{m}$	mass flowrate, kg/s
M_x	moment measured by wind tunnel balance on x axis, net of tare load, Nm
M_y	moment measured by wind tunnel balance on y axis, net of tare load, Nm
M_z	moment measured by wind tunnel balance on z axis, net of tare load, Nm
M_p	propeller's pitching moment due to angle of incidence, Nm
N_p	propeller's normal force due to angle of incidence, N
n	propeller's frequency, revolution/s
P	Power, W
$P_{induced}$	propeller's induced power, W
P_{useful}	propeller's useful or propulsive power, W
P_{total}	propeller's ideal or theoretical total power, W
p	static pressure, Pa
p_{atm}	atmospheric pressure, Pa
$pitch$	propeller's geometric pitch, in/revolution
Q	torque, Nm
q	free stream dynamic pressure, $= 0.5\rho V^2$, Pa
RPM	propeller's revolution per minute, rpm
S_{disk}	propeller's swept area or rotor disk area, m^2
S_{eff}	propeller's momentum balance effective area, m^2
S_{wing}	equivalent wing area for T_{wing} calculation, m^2

T	propeller's thrust, N
$\mathbf{T}$	propeller's thrust vector, N
T_{axial}	axial component of thrust, N
T_{wing}	wing lift equivalent component of thrust, N
V	free-stream or wind velocity, m/s
$\mathbf{V}$	free-stream velocity vector, m/s
V_{disk}	slipstream velocity at the rotor disk, m/s
$\mathbf{V_{disk}}$	slipstream velocity vector at the rotor disk, m/s
V_{ult}	slipstream ultimate velocity, m/s
$\mathbf{V_{ult}}$	slipstream ultimate velocity vector, m/s
WF	wing factor
w	propeller's induced velocity at the rotor disk, m/s
$\mathbf{w}$	propeller's induced velocity vector, m/s
w_o	propeller's induced velocity at disk in static condition, m/s
w_{ult}	ultimate induced velocity, m/s

Greek Symbols

α_p	propeller's angle of incidence, deg
α_{slp}	slipstream angle of incidence at the rotor disk, deg
$\alpha_{slp_{ult}}$	slipstream ultimate angle of incidence, deg
$\beta_{0.75}$	propeller's blade pitch angle at 75% radius station, $=$ atan$(\frac{pitch/D}{\pi\, 0.75})$, deg
Δp	pressure differential, Pa
$\partial/\partial\alpha_p$	derivative or sensitivity with respect to AoA, 1/deg
$\partial/\partial V$	derivative or sensitivity with respect to V, s/m
ϵ	angle between $\mathbf{T}$ and $\mathbf{V_{disk}}$, deg
ϵ_{ult}	angle between propeller axial direction and $\mathbf{V_{ult}}$, deg
η	efficiency
Ω	propeller's angular velocity, rad/s
ρ	atmospheric density, kg/m^3
σ	standard deviation

Subscripts

axial	relative to the axial component
wing	relative to the wing component
ult	ultimate or far wake
$\|_{V=0}$	static condition
$\|_{AoA=0}$	no incidence condition

Abbreviations

AR	Aspect Ratio
BEMT	Blade Element Momentum Theory
CAGR	Compound Annual Growth Ratio
CFD	Computational Fluid Dynamics
ESC	Electronic Speed Controller
eVTOL	Electric VTOL
IMU	Inertial Measurement Unit
MAS	Master Airscrew
MEMS	Micro Electro-Mechanical Systems
PVTOL	Planar and VTOL
rhs	right hand side
rmse	root mean square error
STOL	Short Take Off and Landing
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take Off and Landing
V/STOL	Vertical and/or Short Take Off and Landing

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background on V/STOL, PVTOL vehicles

Large interest in V/STOL vehicles development appeared in the late 1940's for several operational requirements, both military and civilian, mainly to avoid the need of large air fields for take-off and landing.

The first heavier than air manned VTOL machine to lift-off was the Breguet brothers gyroplane in 1907, in a one meter height, unstable and uncontrollable brief flight, steadied by four men. Later in 1935, the "Gyroplane Laboratore" by Breguet-Dorand performed the first successful helicopter flight. In 1939 Igor Sikorsky flew the first practical helicopter, the Sikorsky VS-300, in the United States [1].

Helicopters have the advantages of VTOL, are capable of hovering, and are highly maneuverable. However, the rotor-craft configuration is intensely energy demanding as the weight of the aircraft is sustained solely by the spinning rotors. Hybrid VTOL or PVTOL vehicles are convertible airplanes designed to take-off and land vertically, being able to fulfill the helicopter role, while preserving the advantages of higher speed and more economic flight features of the airplane in planar flight mode. The addition of fixed winged aircraft principles to VTOL rotor-crafts can add an improvement in aerodynamics efficiency of around four times [2]. While helicopters typically achieve a lift to drag ratio (L/D) between 4 and 5, a winged hybrid vehicle could achieve an L/D ratio of approximately 20. PVTOL vehicles, therefore, can offer the required range and energetic efficiency improvement for rotor-craft vehicles.

The Convair POGO XFY-1, (Fig. 1.1), a manned "tail sitter" aircraft developed for the US Navy, is cited to be the first VTOL convertible airplane flown in history. The aircraft would take off vertically, perform a transition to forward planar flight and back to hover, before landing [3]. It first flew in 1954, but difficulties for pilot vision in takeoff and landing, and performance complications caused the project to be abandoned. In the subsequent decades after WWII, improvements in helicopter technologies, and advances of the jet age inhibited further research in PVTOL propeller driven aircraft.

During the cold war, interest in V/STOL fighter jets able to operate on aircraft carriers or on short inland runways led to the development, in the late 1950s, of a thrust vectored aircraft prototype that later, in the 1960s, emerged as the Hawker Siddeley Harrier (Fig. 1.2), in a few versions [4]. The Soviet Union also developed its V/STOL version for carriers, the Yak-38, which began flying in 1971 [5, p.12]. In recent years, project specifications for a V/STOL stealth supersonic fighter by the US armed forces, led to the creation of the F-35, in a few variants (see Fig. 1.3). In march 2010, the F-35 completed its first vertical landing.

Figure 1.1: The Convair XFY-1 "Pogo"
Source: www.defensemedianetwork.com/stories/tail-sitter-xfy-1-pogo-xfv-1

Figure 1.2: Sea Harrier FRS Mk2 RN XZ497
Source: www.baesystems.com/en/heritage/hawker-siddeley-harrier

Figure 1.3: An F-35 landing vertically
Source: Lockheed Martin

As jet age technologies evolved, propeller thrusted vehicles still preserved their role and importance, for many applications. The requirement for propeller V/STOL has also constantly increased, for operations in confined urban spaces, in difficult access missions, for rescue, or for economic reasons, when the use of jet power is impracticable. The helicopter, even with its technological advances, is still subjected to limitations in speed, range, load capacity, and fuel consumption. This has renewed the demand and research for more versatile and aerodynamic efficient propeller driven V/STOL aircraft.

The Bell-Boeing V-22 Osprey, a historic innovative PVTOL convertible aircraft, first flew in 1989, (see Fig.1.4). That aircraft combines the vertical performance of a helicopter with the speed and range of a fixed-wing aircraft, by utilizing tilt-rotor

Figure 1.4: The V-22 Osprey
Source: http://www.boeing.com/history/products/v-22-osprey.page

Figure 1.5: The X-2 Defiant
Source: www.lockheedmartin.com/en-us/products/sb1-defiant-technology-demonstrator.html

technology, where large propellers tilt from vertical to horizontal operation mode, when in forward planar flight [6]. The X-2 Defiant, by Sikorsky/Lockheed in partnership with Boeing (Fig.1.5), is being developed as the next generation helicopter. It features a rear pusher propeller and two coaxial contra-rotating rotors, being able to overcome the retreating blade stall effect at high speeds, and therefore, increasing horizontal velocity limits with improved maneuverability, while reducing power consumption in an attempt to approach the performance of an airplane in forward flight [7].

1.2 The surge of electric aircraft technology

In recent years, electric aircraft technologies have been rapidly advancing, with the advent of MEMS (Micro-Electro-Mechanical Systems), and the development of light, small, low cost electronics used in IMUs (Inertial Measurement Units), accompanied by ever increasing improvements in computing power technologies, batteries, and electrical DC motors [8–12]. As a consequence of these new technologies, demand has been rising across the globe for drone applications. There has been increasing utilization of UAVs in agriculture and forestry survey, construction inspections, search and rescue operations, parcels deliveries, police and border surveillance, entertainment, film industries, etc. Lower operational costs associated with lower human risks and versatility are generally the drives for drone utilization [11, 13, 14].

A precise number for drones global market size and growth is unclear, with many different estimates [15–18]. However, all projections account for a large growth in the coming years, with annual growth figures larger than 13 %. One recent survey [16] has indicated that the global military drone market size is projected to reach US$ 23.7 billion in the next 6 years, from US$ 10.53 billion in 2019. In another report [19], it is mentioned that the global commercial drone market is expected to reach US$ 40 billion by 2024, a tenfold growth from the US$ 4 billion market in 2018.

A wide range of organizations and industries are involved in this market, including governments, the military, and new startups and innovators, that are to benefit from expected new drone-supportive regulations by government bodies worldwide. Currently, several eVTOL propeller thrusted projects are under way, for unmanned vehicles, and also intended for manned flight, (see Ref. [20]). In many cases, hybrid vehicle designs are chosen to include the wing feature in order to increase performance, in terms of range, speed, endurance, and payload size, as compared to other battery-powered configurations. Many of these design configurations will employ some form of propeller tilt operation from vertical to horizontal attitude, as in the tilt-wing, where wing and propellers tilt in tandem, the tilt-rotor, where only propellers tilt, or the tail-sitter, where the whole aircraft performs the tilt manoeuvre. It is expected [17] that the global demand for hybrid PVTOL drones will experience the highest

CAGR over other configuration designs, in the coming years.

Figure 1.6 shows NASA's GL-10 prototype, a tilt-wing V/STOL vehicle concept being developed to take advantage of propeller slipstream for enhancement of wing performance, through distributed electric propulsion along the wingspan [21]. The increased lift is caused by the rise in dynamic pressure on the wings, due to the propellers. At low speeds, it will also allow for shorter runway landings. Figures 1.7, and 1.8 present two different eVTOL drone configurations for parcels delivery. A tilt-wing UAV is used by the German company DHL [22] in the first case, while the second utilizes a tilt-rotor version, manufactured by German maker Wingcopter [23]. Figure 1.9 depicts the Swiss tail-sitter Wingtra [24], for photogrammetry surveying.

Figure 1.6: NASA's GL-10 tilt-wing prototype. Source: Rothaar et al [21]

Figure 1.7: DHL "Parcel-Copter" tilt-wing. Source: DHL/Andreas Heddergott [22]

Figure 1.8: German Wingcopter tilt-rotor. Source: Wingcopter [23]

Figure 1.9: Swiss Wingtra tail-sitter. Source: Wingtra [24]

Figure 1.10: The American Joby Aviation, tilt-rotor air-taxi for pilot and four passengers. Source: www.jobyaviation.com

Fig. 1.10 depicts an electric tilt-rotor air-taxi being developed in the USA, by Joby Aviation [25]. Countless other projects worth mentioning, (see Ref. [20]), are not cited here for the sake of brevity.

1.3 Thesis Motivation

The rapidly expanding markets for propeller driven hybrid PVTOL vehicles are creating project opportunities, globally. Several design projects employ concepts such as the tilt-wing, tilt-rotor, tail-sitter, or variations of those, which require propellers to perform at flight angles varying from zero to around 90^o. The transition between these flight modes in tilt-mode vehicles is mostly brief. Nonetheless, understanding and modelling propellers behavior under a wide range of angles is crucial to determine these vehicles performance.

The behaviour of propellers at intermediate angles has been briefly studied during the early years of PVTOL propeller vehicle developments at the dawn of the jet age, when research was mostly experimental and limited [26–28], with only a few complicated models proposed [29, 30]. Nowadays, analysis at incidence is done through complex CFD models and/or through experimental methods.

In general, propellers are projected to operate either as airplane propellers at near-axial conditions, where the oncoming airflow direction is nearly parallel to the propeller rotation axis, or designed as helicopter rotors, where the airflow is nearly

aligned with the rotor disk plane. Propeller design projects require complex and costly, time consuming computational models and experimental evaluation, for high precision results.

For preliminary PVTOL vehicle projects, employing complex computational models required for propellers design, might not be the adequate approach for a simplified propeller thrust analysis, or for when existing off-the shelf propellers are used, as is the case in many current small scale PVTOL vehicle projects. The need for a simpler, straightforward thrust model, that accounts for the thrust performance at angles of incidence, is the desired solution to understand the general thrust behaviour of the propeller, in many cases.

1.4 Thesis Objectives and Research Question

In order to avoid complex analysis and costly computational efforts for PVTOL vehicle initial designs, this research aims at understanding the general thrust behaviour at angles of incidence and, possibly to develop a simple yet relatively accurate model for the thrust performance of the general propeller regardless of its geometry. A further interest is to investigate the specific AoA conditions that, at high speeds, cause the propeller to behave as a wing (see [31, p.319], [32]), and understand the underlying physical mechanism involved in the process. In that sense, a question arises: "Would it be possible to characterize and specify such conditions at which the wing-like behaviour starts ruling over the general performance of the propeller?"

1.5 Thesis Structure

The thesis is structured as follows:

- **Chapter 1** The current introduction chapter presented a brief background, the motivation and objectives of the thesis.

- **Chapter 2** presents a literature review of propellers modelling, emphasizing the Classical Actuator Disk theory for propellers at angles of incidence, the theoretical method applied in this research, chosen for its simplicity and yet relatively

accurate thrust results [33, 34]. The mathematical development of the theory is also provided.

- **Chapter 3** offers a new model development based on the Classical Momentum Theory and on Glauert's Hypothesis [31], where by considering an enhanced mass flow rate through the disk, thrust is mathematically decomposed into two parts: T_{axial}, which is dependent on the axial component of the oncoming wind speed, and T_{wing}, which is sensitive to the wind component parallel to the rotor plane. The development quantifies the contribution of each component to total thrust and helps clarify, under the scope of momentum theory alone, why rotors in forward flight behave as wings and why thrust increases with AoA.

- **Chapter 4** describes the general experimental wind tunnel methods applied in order to validate the theoretical findings. The wind tunnel tests were performed at angles of incidence ranging from 0° to 90°.

- **Chapter 5** presents the initial wind tunnel tests campaign results and analysis. The effects of RPM, AoA, and oncoming flow velocity V on thrust measurements are experimentally evaluated for a small scale 2-bladed fixed-pitch propeller. The propeller's behavior is explained through the influence of the two thrust components. Discussion on slipstream parameters, in response to AoA, is also provided. Finally, under certain assumptions, a simplified formula is derived to allow for the estimation of the propeller's thrust when at incidence, based on the performance data measured at $AoA = 0^o$.

- **Chapter 6** describes the results and analysis of the subsequent tests campaign performed on four different small scale fixed-pitch propellers, with emphasis on the relative influence of each component on the general thrust behaviour at high angles of incidence. The tests results show that, under certain circumstances, propellers begin to behave as wings. These conditions are explained, in light of the new theory.

- **Chapter 7** summarizes key findings of the research, and suggestions for further work are proposed.

Chapter 2

Overview of Propellers Modelling

2.1 Literature Review

In the past century, several models to analyse propellers thrust behaviour with different accuracies and complexities were developed. The simplest model to assess propeller thrust performance is the Classical Momentum Theory or Actuator Disk Model. It was first introduced for marine propellers by Rankine in 1865 [35] and Froude [36] in 1889. Later, in the beginning of the 20th century, it was adopted for airscrews with the advent of the airplane. The theory applies the basics of fluid dynamics conservation laws - mass (continuity), momentum and energy - to provide a general understanding of the propeller performance. Its assumptions disregards the propeller blades' geometry and details of the flow about them. The propeller is considered as an infinitely thin actuator disk that impels a sudden increase in pressure on the fluid as it flows across its surface. This causes an acceleration of the flow, so an induced velocity increment is modelled at the disk. The flow is assumed to be incompressible, inviscid, and rotation is neglected. Also, the velocity and pressure over the disk are considered to be uniform and so thrust is distributed evenly over its surface. The major limitations of the theory are that it does not take profile drag losses of the propeller blades into account, nor blades' tip vortices and rotation effects. Despite the simplifications assumed, the patterns of velocities and pressures in the actuator disk model have been verified experimentally [37] and although it is not very accurate for power estimates [33], it provides a very good approximation for thrust. The theory is, however, not expected to yield a good basis for rotor in plane

analysis and hence it is not suitable for propeller design by itself. The derivation of the equations involved in the Classical Axial Momentum Theory for propellers can be found in several sources [31, 38, 39].

Subsequent models with higher complexity arose in the wake of the Actuator Disk Theory. General or Extended Momentum Theory, where the incorporation of rotation is added to the model, was developed by Betz in his work "An extension of the screw beam theory" [40] (Betz, 1920). The Blade Element Theory (BET), that considers the geometry of the blades, was first devised by Drzewiecki, [41], in 1920. It ignored the effect of the induced flow inside the streamtube [42], as defined in the Momentum Theory, taking into account only the freestream velocity V and the propeller rotation for every blade element analysis. The more precise Blade Element-Momentum Theory (BEMT) incorporated the induced velocity. Other variations of BEMT calculate the induced velocity in different ways. Goldstein's Classical Vortex Theory [43] related the induced velocity to the bound circulation around every blade element. Vortex theory studies were later enhanced by Theodorsen [44], in 1948. An extensive compendium of rotor aerodynamics analysis by Joukowsky in light of momentum theories is presented in [45] and under vortex theories in [46].

Previous relevant studies to incorporate non-axial flow conditions were made by Ribner [29, 47], who devised formulas, utilizing BEMT, to calculate side forces and moments that appear when propellers operate at incidence. The development assumed low angles of incidence. Young in 1965, [30] later modified Ribner formulas and expanded the use for high angles and also analyzed the effects of incidence on thrust. As mentioned before in Chapter 1, experimental investigations have also been made in order to understand the behavior of full-scale propellers at a wide range of incidence in [26–28], where it has been shown that thrust increases with increasing AoA, and that thrust generally grows with the advance ratio J at high AoA, as opposed to the effect at low angles. Recently, experiments on several small scaled low-pitch propellers for UAV applications [48, 49] also showed that, for $AoA < 60^o$ C_T diminishes with the advance ratio J, and vice-versa for higher angles of incidence. It is worth mentioning also a few studies on wind turbines under yaw conditions,

presented in [50–57]. This list could be expanded greatly, therefore it should not be considered exhaustive.

Glauert [31, p.319], [32] conducted an analysis on helicopter rotors in forward flight utilizing Momentum Theory. Although it is mentioned that no solid proof has been given, he introduced a thrust formula for angles of incidence that reverts to the classical static thrust equation in the case of zero forward speed and, at the other extreme at fast forward flight, the equation takes the form of the elliptic wing lift formula, implying that the rotor behaves as a wing in that case. Bramwell [58] investigated the validity of Glauert's formula by solving the linearized Euler's equations with small disturbances in the velocity field, showing that, for lightly loaded propellers, the model is valid for the axial case and for high speeds at 90°, regardless of blade geometry. Moreover, in the latter case, the linear theory "appears" to hold also for non-uniform load distributions. Glauert's Hypothesis for propellers at incidence has been widely utilized also for a wide AoA range [33, 37, 38, 42].

The appeal of the Actuator Disk Momentum theory simplicity attracted further investigation to determine its limitations and possible improvements of its accuracy. Van Kuik [59, 60] mentions that the average induced velocity calculated with the assumptions of the classical theory is underestimated. For a uniform load, a singularity at the disk edge must be included through a correction represented by a discrete vortex carrying an edge force. Goorjian [61] also states an inconsistency in the General Momentum Theory, as mutual interferences between different annular elements are not considered. The existence of this inconsistency was also known to Glauert, but only in recent years the errors associated with it have been quantified. This is due to Sørensen and Mikkelsen [62], Van Kuik and Lignarolo [63] and Bontempo and Manna [64] for the Axial Momentum Theory, and in the general case, to Sørensen [65], and Bontempo and Manna [66, 67]. Conway [68] developed an analytical closed solution for the linearized actuator disk with arbitrary radial load distributions. As an extension of the linearized solution, a semi-analytical method was then developed for a non-uniform heavily loaded disk in [69]. Pitt and Peters [70] presented a linear, unsteady actuator model considering the dynamic inflow behavior of helicopter ro-

tors. A model for axial and for skewed flows by Morillo and Peters [71] presents the solution of the complete velocity field above the disk by converting the potential flow equations into ordinary differential ones. Recently, Rosen and Gur [72] developed an axisymmetric axial actuator disk model including radial and tangential induced speed components. The model defines a pressure ratio factor that depends on the blades geometry to conclude the momentum balance, for which calculations are performed iteratively. Later, Kominer and Rosen [73] adapted the model for asymmetric skewed inflows.

Although the previously cited Actuator Disk Models improve accuracy, many of them are cumbersome to implement and end up missing the advantage of the Momentum Theory simplicity. For high accuracy, alternative investigation approach for propellers at angles of incidence uses computational fluid dynamics (CFD) to solve the 3D flow equations. These models may include viscosity, non-uniform propeller inflows, complex geometry and more complicating assumptions such as the addition of nacelles and wing interference in the analysis [48, 74, 75]. Often times, high complexity CFD models are coupled with experimental tests for validation. In general, they are also computationally expensive and time consuming to implement.

2.2 The Classical Momentum Theory or Actuator Disk Model for Propellers at Incidence

2.2.1 Introduction

The classical momentum theory models an inviscid flow, with a uniform pressure jump and an average uniform induced speed at the actuator disk. Rotation is neglected.

To apply the principles of momentum theory at incidence, it is first necessary to define the mass flow rate through the disk and the boundaries of the streamtube. As the oncoming airflow velocity reaches the propeller disk at an angle, initially one would assume the normal component of the velocity to the disk plane to estimate the mass flux through the disk area. As for the boundaries of the streamtube, the classical axial momentum theory models it touching the disk rim. However, it has

been determined experimentally that the rotor entrains air from outside the rim streamtube [38, p.123], a fact that ever increases with the incidence. Shapiro [37, p.17] also postulates the hypothesis of a much larger region of induced flow affected by the rotor at incidence and assumes a wider streamtube, where the mass of air that takes part in the exchange of momentum is the mass flux through a projected area normal to V_{disk} and equivalent to S_{disk}, regardless of the angle of incidence.

The phenomenon can be explained to be caused by radial pressure gradient effects not modelled in the classical momentum theory. The radial pressure gradients cause an increase in mass flow that grows with AoA, by means of tip vortices. A rigorous description of the phenomenon can be found in [59–62]. As the radial momentum balance is disregarded, the uniform pressure load at the disk would cause an infinite radial pressure gradient at the edge and a velocity singularity, which are inconsistent. This inconsistency in the model can only be resolved through the addition of edge vortices, that act as "natural concentrators" (see Refs. [59, 76]), which increase the mass flow through the disk. This effect grows with AoA and with the speed component in the rotor plane (see ref.[73]).

2.2.2 The Theoretical Mass Entrainment Factor e

In order to incorporate an increase in mass flow rate to the model, an entrainment factor e is applied to the disk area and a new effective area $S_{eff} = e\, S_{disk}$. Then, $\dot{m}$ can be expressed as:

$$\dot{m} = \rho \left(V \cos \alpha_p + w \right) S_{eff} \tag{2.1}$$

The wider effective streamtube assumed can be defined by flipping the rotor disk area to a normal position relative to V_{disk} (see Fig. 1). The mass flow rate in the effective streamtube would then be given as:

$$\dot{m} = \rho\, V_{disk}\, S_{disk} \tag{2.2}$$

Now, stretching the disk area at the original position to reach the new streamtube boundaries will define a new enhanced or effective disk area S_{eff} that is adopted for the mass flux calculation in Eq. (2.1). The following development is based on the

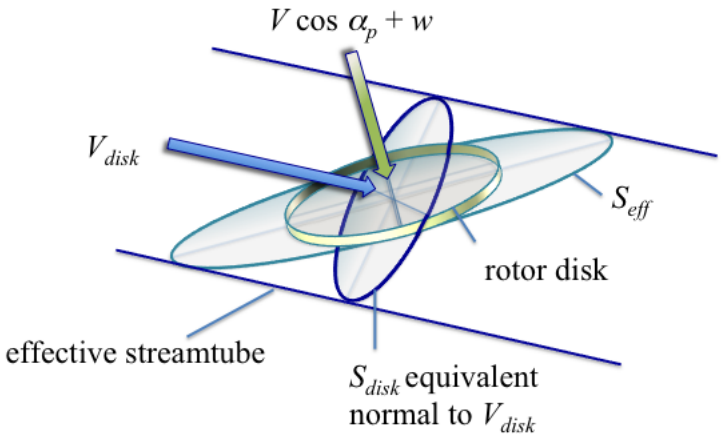


Figure 2.1: Effective streamtube defined by an equivalent area S_{disk} normal to V_{disk}

assumption that Eqs. (2.1) and (2.2) are equivalent for the definition of the mass flowrate $\dot{m}$. This concept that determines the mass flowrate through a circle, with an area defined by the propeller diameter and normal to V_{disk}, comes from rotor analysis in forward flight and uses the analogy to the wing theory, see ([77, p.117], [31, p.319], [39, p.125], [37, p.16]). It was presented in Glauert Hypothesis, [31, p.319].

Figure 2.2 illustrates the scheme of the velocity vectors for propellers at incidence. The free stream velocity projected on the propeller's reference frame can be shown as:

$$\mathbf{V} = -V\cos\alpha_p\,\widehat{\mathbf{e}}_{\mathbf{T}} + V\sin\alpha_p\,\widehat{\mathbf{e}}_{\mathbf{N}} \tag{2.3}$$

By following the momentum theory, where the velocity at the disk is the vector sum of the axial induced velocity by the propeller and the free-stream speed:

$$\mathbf{V_{disk}} = \mathbf{V} + \mathbf{w} \tag{2.4}$$

from which:

$$\mathbf{V_{disk}} = -(V\cos\alpha_p + w)\,\widehat{\mathbf{e}}_{\mathbf{T}} + V\sin\alpha_p\,\widehat{\mathbf{e}}_{\mathbf{N}} \tag{2.5a}$$

$$|V_{disk}| = \sqrt{(V\cos\alpha_p + w)^2 + (V\sin\alpha_p)^2} \tag{2.5b}$$

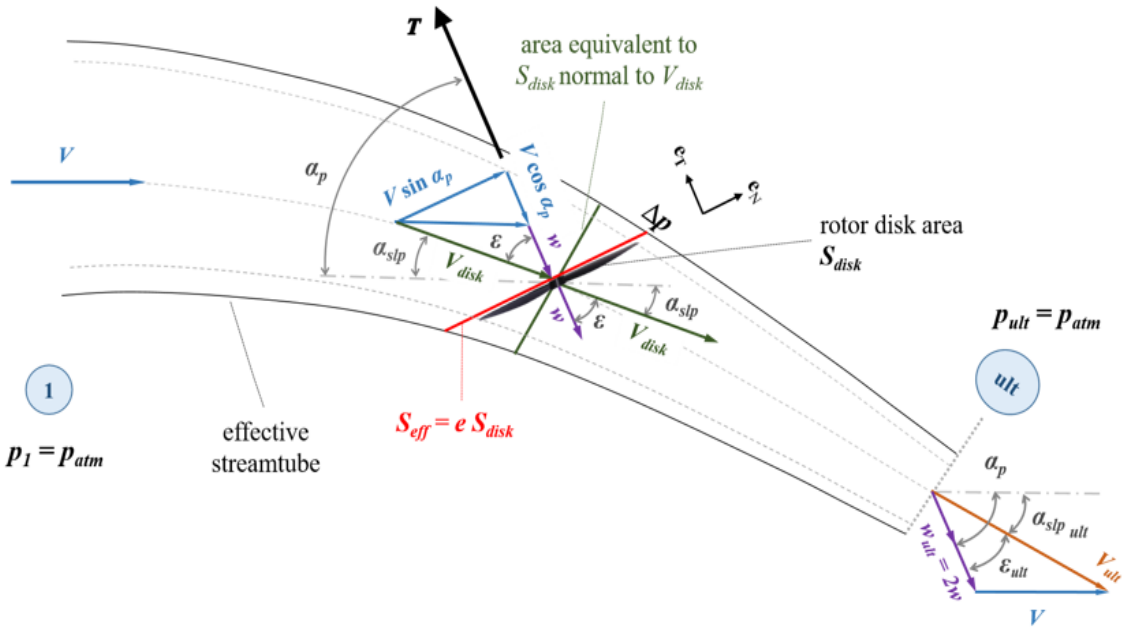


Figure 2.2: Illustration of velocity vectors for propellers at incidence

Now, from vector geometry in Fig. 2.2 it follows that:

$$V_{disk}\ \cos\epsilon = V\cos\alpha_p + w \tag{2.6}$$

then, by substituting Eqs. (2.2) and (2.6) into Eq. (2.1) leads to the determination of the theoretical mass entrainment factor e as:

$$e = \frac{V_{disk}}{(V\cos\alpha_p + w)} = \sqrt{1 + \frac{(\sin\alpha_p)^2}{(\cos\alpha_p + w/V)^2}} = \frac{1}{\cos\epsilon} \tag{2.7}$$

which leads to the relation between the effective area and the disk area to be:

$$S_{eff} = \frac{S_{disk}}{\cos\epsilon} \tag{2.8}$$

Fig.2.3 illustrates the variation of the entrainment factor e determined by using Eq. (2.7) with ϵ. The effects of AoA and w/V on e is shown in Fig.3.1. It can be seen that at low angles of incidence, e tends to one. The effective area S_{eff} is similar

to the rotor disk area, as the propeller is operating at air speeds near axial condition. S_{eff} is increased with e and therefore with AoA. As AoA approaches 90° and $V \cos \alpha_p$ tends to zero, e tends to infinity in theory at very low w/V, so as to increase the area S_{eff} in order to maintain the same finite value of mass flow rate $\dot{m}$ through the rotor, according to Eqs.(2.1), and (2.2). In these extreme cases, the angle ϵ approaches 90°, as V_{disk} is almost aligned with the rotor disk. The increase in e becomes more relevant at high angles of incidence and at very low w/V values.

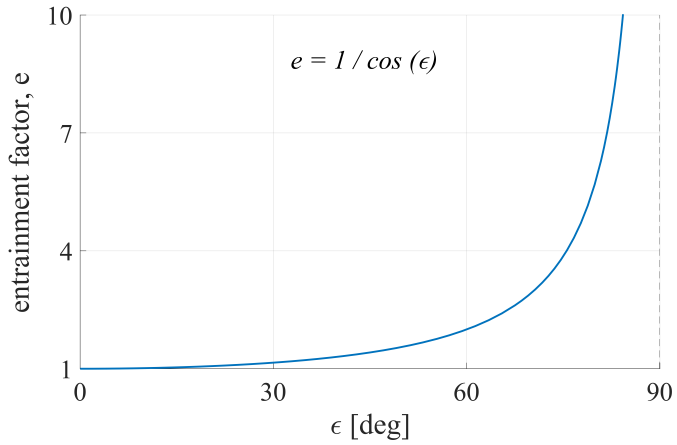


Figure 2.3: Variation of e as function of ϵ, determined from Eq. (2.7)

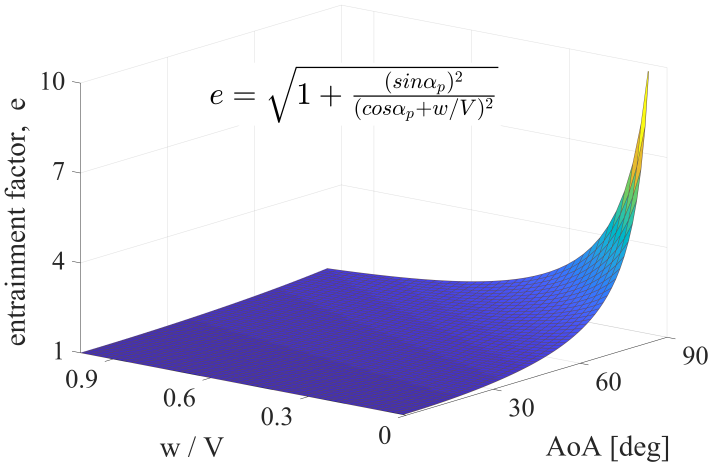


Figure 2.4: Variation of e as function of AoA and w/V, determined from Eq. (2.7)

2.2.3 Thrust and Geometric Characteristics of the Slipstream

Having the entrainment factor e and the mass flow rate predicted, we now consider the momentum balance in the propeller axis direction $\widehat{\mathbf{e}}_{\mathbf{T}}$, between the ultimate wake section far downstream in the streamtube and section 1 far upstream of the propeller in Fig.2.2. Recalling that the classical momentum theory disregards the wake rotation effects and assumes the flow to be inviscid, then thrust is the only force imparted to the flow and can be written as:

$$T = \dot{m}\,(V_{ult}\,\cos\epsilon_{ult} - V\cos\alpha_p) \tag{2.9}$$

from geometric relations in Fig. 2.2, it is possible to see that:

$$V_{ult}\,\cos\epsilon_{ult} = V\cos\alpha_p + w_{ult} \tag{2.10}$$

which leads to:

$$T = \dot{m}\,w_{ult} \tag{2.11}$$

Now, the substitution of Eq. (2.1) or, alternatively Eq. (2.2) into Eq. (2.11) allows thrust to be rewritten as:

$$T = \rho\,(V\cos\alpha_p + w)\,S_{eff}\,w_{ult} \tag{2.12a}$$

$$T = \rho\,V_{disk}\,S_{disk}\,w_{ult} \tag{2.12b}$$

Considering Bernoulli along the streamlines of the streamtube between section 1 and immediately prior to the disk (-) and from immediately after the disk (+) to the ultimate wake we get the following two equations:

$$p_1 + \frac{1}{2}\rho V^2 = p_{disk}^{-} + \frac{1}{2}\rho {V_{disk}^{-}}^2 \tag{2.13a}$$

$$p_{ult} + \frac{1}{2}\rho V_{ult}^2 = p_{disk}^{+} + \frac{1}{2}\rho {V_{disk}^{+}}^2 \tag{2.13b}$$

But $V_{disk}^{-} = V_{disk}^{+} \triangleq V_{disk}$ for continuity through the disk to hold, and $p_1 = p_{ult} = p_{atm}$. At the disk, the theory models a jump in pressure Δp impelled by the propeller, therefore: $p_{disk}^{+} = p_{disk}^{-} + \Delta p$. Considering all these equalities and subtracting one equation from the other, we end up with:

$$\frac{1}{2}\rho\,(V_{ult}^2 - V^2) = \Delta p \tag{2.14}$$

where V_{ult} can be obtained from the illustration in Fig.2.2 as:

$$V_{ult}^2 = (V + w_{ult}\ \cos\alpha_p)^2 + (w_{ult}\ \sin\alpha_p)^2 \tag{2.15}$$

Substituting Eq. (2.15) into Eq. (2.14) and considering that the thrust is due to the pressure jump at the disk, then:

$$\Delta p = \frac{T}{S_{eff}} = \frac{1}{2}\rho\left[(V + w_{ult}\ \cos\alpha_p)^2 + (w_{ult}\ \sin\alpha_p)^2 - V^2\right] \tag{2.16}$$

Substituting Eq. (2.12a) into Eq. (2.16) and expanding terms in the right hand side leads to:

$$\rho\left(V\cos\alpha_p + w\right) w_{ult} = \frac{1}{2}\rho\left(w_{ult}^2 + 2w_{ult}\,V\cos\alpha_p\right) \tag{2.17}$$

then solving algebraically yields:

$$w_{ult} = 2\,w \tag{2.18}$$

and so thrust can be expressed, by using Eqs. (2.8) and (2.18) into Eq. (2.12a), as:

$$T = 2\rho\left(V\cos\alpha_p + w\right) e\,S_{disk}\,w \tag{2.19}$$

On the other hand, T can also be rewritten from Eqs. (2.18) on Eq. (2.12b) as:

$$T = 2\rho\,V_{disk}\,S_{disk}\,w \tag{2.20}$$

which is the well-known equation of thrust from Glauert's hypothesis in [31, p.319], [39, p.213]. Replacing Eq. (2.5b) into (2.20) leads to a quartic equation in w as:

$$\left(\frac{T}{2\rho\,S_{disk}}\right)^2 = V^2w^2 + 2Vw^3\cos\alpha_p + w^4 \tag{2.21}$$

As thrust can be expressed as $T = \rho n^2 D^4 C_T$ and the advance ratio of the propeller is defined as $J = V/nD$, then Eq. (2.21) can be rewritten as:

$$\left(\frac{2n^2D^2C_T}{\pi}\right)^2 = V^2w^2 + 2Vw^3\cos\alpha_p + w^4 \tag{2.22a}$$

$$\left(\frac{2C_T}{\pi J^2}\right)^2 = \left(\frac{w}{V}\right)^2 + 2\left(\frac{w}{V}\right)^3\cos\alpha_p + \left(\frac{w}{V}\right)^4 \tag{2.22b}$$

As per Eq. (2.22a), an increase in the propeller frequency n, or in RPM, is associated with an increase in w. The same effect can be observed in terms of J from Eq. (2.22b), where a raise in J, or an increase in V relative to n or RPM, will cause a decrease in w/V and, vice-versa, a rise in the ratio of the propeller rotation to wind speed V will incur in an increase in w/V. For the particular case of static thrust, where $V = 0$, one obtains the classical induced velocity for hover condition $w|_{V=0}$.

$$w|_{V=0} = \sqrt{\left(\frac{T}{2\,\rho\,S_{disk}}\right)} \tag{2.23}$$

For $AoA = 0^o$, (i.e. no incidence), and if $T > 0$, the only physically valid 4 root solution of the Eq. (2.21), is as below (see Ref. [33, p.345]):

$$w|_{AoA=0^o} = \frac{1}{2}\sqrt{\left(\frac{2T\,|_{AoA=0^o}}{\rho\,S_{disk}}\right) + V^2} - \frac{V}{2} \tag{2.24}$$

which can be rewritten as:

$$\left(\frac{w}{V}\right)_{|_{AoA=0^o}} = \frac{1}{2}\sqrt{\left(\frac{8C_{T\,|_{AoA=0^o}}}{\pi J^2}\right) + 1} - \frac{1}{2} \tag{2.25}$$

The derived equations from the classical momentum theory for propellers at a given angle of incidence permits the estimation of the theoretical slipstream angle at the disk α_{slp} and at the ultimate wake $\alpha_{slp_{ult}}$. As illustrated in Fig.2.2, it can be shown from Eqs. (2.15), and (2.18) that:

$$V_{ult} = V\sqrt{1 + 4\left(\frac{w}{V}\right)\cos\alpha_p + 4\left(\frac{w}{V}\right)^2} \tag{2.26}$$

$$\alpha_{slp_{ult}} = \arcsin\left(\frac{2\left(\frac{w}{V}\right)\sin\alpha_p}{\sqrt{1 + 4\left(\frac{w}{V}\right)\cos\alpha_p + 4\left(\frac{w}{V}\right)^2}}\right) \tag{2.27}$$

and at the disk, α_{slp} can be shown to be:

$$\alpha_{slp} = \alpha_p - \epsilon \; = \; \alpha_p - \arctan\left(\frac{\sin\alpha_p}{\cos\alpha_p + \left(\frac{w}{V}\right)}\right) \tag{2.28a}$$

$$\alpha_{slp} = \arcsin\left(\frac{\left(\frac{w}{V}\right)\sin\alpha_p}{\sqrt{1 + 2\left(\frac{w}{V}\right)\cos\alpha_p + 2\left(\frac{w}{V}\right)^2}}\right) \tag{2.28b}$$

It has been shown that the influence on thrust by the entrainment factor e is correlated to the angle ϵ. From Fig. 2.2 it is easy to see that:

$$\tan\epsilon = \frac{V\ \sin\alpha_p}{V\ \cos\alpha_p + w} \tag{2.29}$$

from where it can be seen that the angle ϵ grows with AoA and V. It is also worth noting that as ϵ grows, (with e), the influence of the oncoming wind velocity component aligned with the rotor plane $V\sin\alpha_p$ increases relatively to the influence of the axial velocity components $V\cos\alpha_p + w$. For no incidence flight, ϵ equals α_p at 0^o. For $\alpha_p = 90^o$:

$$\tan\epsilon = \frac{V}{w} \tag{2.30}$$

2.2.4 Power and Efficiency

From energy balance, the total input power imparted to the fluid by the rotor, is equal to the power causing a change in the kinetic energy of the flow passing through the propeller:

$$P_{total} = \frac{1}{2}\dot{m}(V_{ult}^2 - V^2) \tag{2.31}$$

which is the work per unit time performed by the propeller, and written as:

$$P_{total} = TV_{disk}\cos\epsilon \tag{2.32}$$

This is the ideal power, or the theoretical power required by a rotor in forward flight, according to momentum theory, where losses are disregarded, [39, p.126]. Substituting Eq. (2.6) on Eq. (2.32) leads to:

$$P_{total} = T(V\cos\alpha_p + w) \tag{2.33}$$

from where it is seen that the total ideal power can be divided in two parts:

a) The propulsive or useful output power, being defined as the product of thrust T and the displacement per unit time, in the thrust direction, [38, p.125], [39, p.126], [33, p.345], and written as:

$$P_{useful} = TV\cos\alpha_p \tag{2.34}$$

b) The induced power that is the power induced by the propeller to the fluid, to produce the change in momentum, and defined as:

$$P_{induced} = Tw \tag{2.35}$$

The theoretical efficiency, is defined as the ratio of the propulsive power, to total power, or the ratio of useful output power to theoretical input power.

$$\eta = \frac{TV\cos\alpha_p}{T(V\cos\alpha_p + w)} = \frac{1}{1+\frac{w}{V\cos\alpha_p}} \tag{2.36}$$

The momentum theory efficiency or ideal efficiency is the maximum theoretical efficiency attainable by the propeller, as rotor profile drag losses and any energy lost in slipstream rotation are disregarded [38, p.125].

Efficiency can also be expressed in terms of the ultimate velocity component. Recalling Eqs. (2.31) and (2.34), yields:

$$\eta = \frac{P_{useful}}{P_{total}} = \frac{TV\cos\alpha_p}{\frac{1}{2}\dot{m}(V_{ult}^2 - V^2)} \tag{2.37}$$

It follows that, substituting T from Eq. (2.9) on Eq. (2.37), yields:

$$\eta = \frac{(V_{ult}\cos\epsilon_{ult} - V\cos\alpha_p)V\cos\alpha_p}{\frac{1}{2}(V_{ult}^2 - V^2)} \tag{2.38}$$

that results in:

$$\eta = \frac{2\cos\alpha_p\left(\frac{V_{ult}\cos\epsilon_{ult}}{V\cos\alpha_p} - 1\right)}{\left(\frac{V_{ult}}{V} - 1\right)\left(\frac{V_{ult}}{V} + 1\right)} \tag{2.39}$$

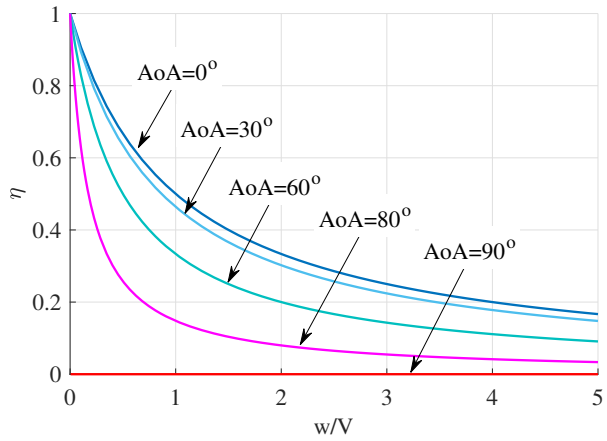


Figure 2.5: Efficiency, η, according to Eq. (2.36), from momentum theory, for different angles of incidence

For the case of no incidence, $AoA = \epsilon_{ult} = 0^o$, then η simplifies to:

$$\eta = \frac{2}{\left(1 + \frac{V_{ult}}{V}\right)} \tag{2.40}$$

which is also a traditional axial momentum theory expression for efficiency.

The effects of w/V and AoA on η are illustrated in Fig. 2.5. From Eq. (2.36) it follows that at hover, when $V = 0$, $w/V \to \infty$, the efficiency is zero, as the propulsive power is zero. The maximum efficiency, according to momentum theory, occurs at $AoA = 0^o$ and at $V \to \infty$. In practice, it never happens, due to rotor blades viscous and tip vortex losses. In addition to the ideal power, more power is required to account for the losses, and also, as V increases at low AoA, the relative angles of the flow on the blades will cause the reduction and ultimate halt of lift production, where $T \to 0$, when also efficiency declines to zero. As V continues to increase, the propeller starts to operate in windmill state. In the case of high AoA, for helicopter rotors, as V increases, there will be a relative wind increase for the advancing blades, however, for the retreating blades, the high V will cause a reduction in relative velocity which eventually will lead to stall and loss of lift production, and therefore, of thrust and efficiency. In any case, as AoA grows, efficiency is reduced as the alignment of T and V diminishes and tends to zero at $AoA = 90^o$.

2.3 Concluding Remarks

In this chapter, an overview of different models to analyse the behaviour of propellers was presented, with emphasis on the development of the Classical Momentum Theory for operations at angles of incidence. This will be the basis for the new development, presented in the next chapter, that will serve as the theoretical methodology, applied throughout this work.

Chapter 3

New Development of the Classical Actuator Disk Model for Propellers at Incidence

3.1 The Expansion of the Entrainment Factor

The following development was first presented in [78]. The entrainment factor e calculated through Eq. (2.7) in Chapter 2 can be expanded as a MacLaurin series as a function of ϵ to yield:

$$e = \frac{1}{\cos\epsilon} = 1 + \frac{\epsilon^2}{2} + \frac{5\epsilon^4}{24} + \frac{61\epsilon^6}{720} + \frac{277\epsilon^8}{8064} + \frac{50521\epsilon^{10}}{3628800} + ... \tag{3.1}$$

then, by naming the second and higher terms of the series as $\sum terms(\epsilon)$, it can be expressed as:

$$e = \frac{1}{\cos\epsilon} = 1 + \sum terms(\epsilon) \tag{3.2}$$

Equation (3.1) shows that e tends to infinity as ϵ approaches 90^o. This effect is captured through the Eq.(3.2) by $\sum terms(\epsilon)$, which is a divergent series with ϵ as it grows to 90^o.

Applying the results of Eq.(3.2) to replace the entrainment factor on Eq.(2.19), will lead to the equation of thrust to be decomposed in two parts as follows:

$$T = 2\rho\left(V\cos\alpha_p + w\right) S_{disk}\, w + 2\rho\left(V\cos\alpha_p + w\right) S_{disk}\, w \sum terms(\epsilon) \tag{3.3}$$

It can be noted that the first component of Eq. (3.3) is independent of ϵ as it arose from the first term of the rhs of Eq. (3.2) that is 1, while the second component,

which bears the summation of divergent terms in Eq. (3.2), is dependent and grows with ϵ.

3.2 Defining Axial and Wing Equivalent Components of Thrust

Recalling Eq. (2.29) in Chapter 2, where it has been noticed that at the disk, ϵ captures an axial component of the velocity, i.e. in the thrust direction $V\cos\alpha_p + w$, and a second component that is the velocity projection on the rotor disk plane, $(V\sin\alpha_p)$.

The expansion in series of the entrainment factor e performed in the previous section demonstrated that the theoretical thrust can be decomposed in two parts, and that the first is independent of ϵ, while the second component grows with ϵ and, hence, as it has been seen, with AoA and V. This fact brings the idea that if one could relate the second term of thrust to be proportional to $\sin\alpha_p$, then that term would capture the influence of the rotor plane velocity projection in an isolated manner from the axial wind component.

Now, it is easy to notice that the axial wind component is captured in the first term of the thrust equation (3.3). That component has exactly the same form as the thrust equation in the axial momentum theory, with only the incoming velocity reduced by a multiplying factor of $\cos\alpha_p$. Therefore we will name the first part as the Axial Component of Thrust, T_{axial}.

$$T_{axial} = 2\rho\left(V\cos\alpha_p + w\right)S_{disk}w \tag{3.4}$$

Now, back to the analysis of the second term, we can observe that, based on Fig. 2.2 and on Eq. (2.29):

$$\frac{V\sin\alpha_p}{V\cos\alpha_p + w} = \tan\epsilon = \frac{\sqrt{1-(\cos\epsilon)^2}}{\cos\epsilon} \tag{3.5}$$

which after a simple manipulation results in:

$$V\cos\alpha_p + w = \frac{V\sin\alpha_p}{\sqrt{\left(\frac{1}{\cos\epsilon}-1\right)\left(\frac{1}{\cos\epsilon}+1\right)}} \tag{3.6}$$

From Eq. (3.2) we get:

$$\sum terms(\epsilon) = \frac{1}{\cos\epsilon} - 1 \tag{3.7}$$

Finally, substituting Eqs. (3.6) and (3.7) onto the second term of Eq. (3.3) allows us to view the second component of thrust under the influence of $V \sin\alpha_p$. After further simple algebraic manipulation, we notice that the term takes the form of the wing lift formula from wing theory, [77, p.117], [31, p.319], [39, p.125], [37, p.16]. Therefore that term will be regarded as the Wing Equivalent Component of Thrust, or T_{wing} and written as:

$$T_{wing} = 2\rho\, V \sin\alpha_p\, S_{disk}\, w \sqrt{\frac{1-\cos\epsilon}{1+\cos\epsilon}} \tag{3.8}$$

Since it is assumed in the actuator disk theory that the induced speed w is constant across the rotor disk, T_{wing} can be considered to be equivalent to the lift produced by an elliptic wing subjected to a wind speed of magnitude $V \sin\alpha_p$ and having the wing surface area equal to:

$$S_{wing} \triangleq S_{disk} \sqrt{\frac{1-\cos\epsilon}{1+\cos\epsilon}} \tag{3.9}$$

3.3 The Propeller Wing Factor

In Eq. (3.9) of the previous section, the factor multiplying S_{disk} and defining the equivalent wing surface area will be regarded as the Wing Factor of the propeller at incidence, WF. It can be expressed also as function of e, as:

$$WF = \sqrt{\frac{1-\cos\epsilon}{1+\cos\epsilon}} = \sqrt{\frac{e-1}{e+1}} \tag{3.10}$$

Substituting Eq. (2.7) on Eq. (3.10) and simplifying it, leads to WF being given as a function of AoA and w/V as well:

$$WF = \frac{\sin\alpha_p}{\sqrt{1 + 2\left(\frac{w}{V}\right)\cos\alpha_p + \left(\frac{w}{V}\right)^2} + \cos\alpha_p + \left(\frac{w}{V}\right)} \tag{3.11}$$

Figure 3.1 shows the variation of WF with the angle ϵ. For low ϵ, WF implies a very small equivalent wing area, vanishing at $\epsilon = 0^o$ (where also $AoA = 0^o$ and the wing component vanishes). For $\epsilon \to 90^o$, WF tends to unity. It yields then a full wing equivalent area of S_{disk}. It can be seen from Fig. 3.2 that WF can only reach unity for $w/V \to 0$, or at high speeds, and at $AoA = 90^o$, which corresponds also to $\epsilon \to 90^o$. For $AoA = 0^o$, WF only vanishes asymptotically as $w/V \to \infty$, when hovering.

In the extreme case, at $AoA \to 90^o$ and $w/V \to 0$, where $WF \to 1$, thrust can

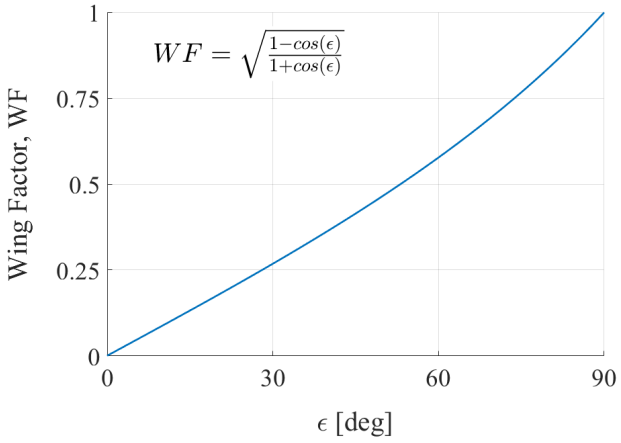


Figure 3.1: Variation of WF with ϵ, as determined by Eq. (3.10)

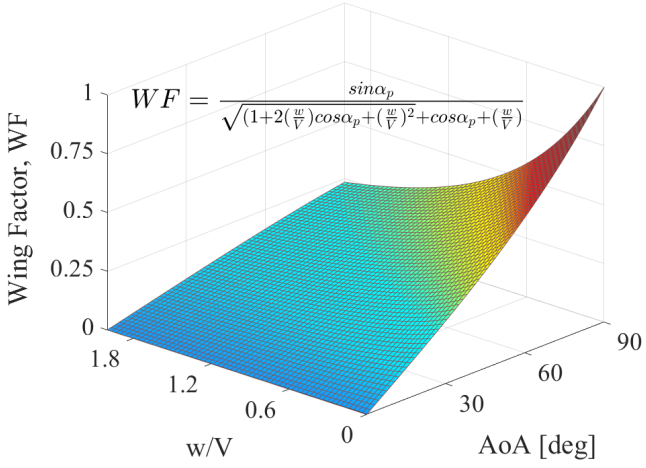


Figure 3.2: Variation of WF with w/V and AoA, as determined by Eq. (3.11)

be shown to be:

$$T = T_{axial} + T_{wing} \rightarrow 2\rho\, S_{disk}\, w^2 + 2\rho\, V\, w\, S_{disk} \tag{3.12}$$

Here, it can be seen that T_{axial} takes the form of the static thrust formula and it contributes to total thrust even at forward speeds. T_{wing} converges to the wing lift formula with the wing equivalent surface area S_{wing} reaching the full area S_{disk} in the case of a rotor at fast forward flight $w << V$, when T_{axial} is negligible. As T_{wing} becomes dominant, the propeller behaves as a wing. These two expressions are mentioned by Glauert, [31, p.319], in the development of his thrust hypothesis in the analysis of helicopters in forward flight. Ariza, [48, p.124], presents results from 3D numerical simulations showing vortices produced by the rotor when at high angles of attack in the same manner as finite wing vortices.

It is suggested that a possible explanation for the phenomenon in those conditions is that the propeller could behave as an elliptic wing. Johnson, [42, p.127], also mentions that the helicopter rotor behaves as a wing at forward speeds and that these two expressions are the limits of thrust when $V >> w$ and when in hover, where $V = 0$. Although he claims that there is no theoretical justification for the approach at intermediate forward speeds, good agreement has been found with measured rotor performance and with vortex theory results, suggesting therefore that it should be accepted for the entire range of speeds. McCormick, [39, p.126], also mentions the analogy of the propeller at forward flight to an elliptic wing when $w/V \rightarrow 0$ and when $V = 0$ to the hover case.

The derivations of Eqs. (3.4) and (3.8) in this work allows us to assume that indeed the two components T_{axial} and T_{wing} are always present and comprise the total thrust for the whole operational envelope, at any angle of incidence and velocity.

3.4 The Wing Equivalent to Axial Component Ratio

In the previous section, the decomposition of thrust in two components has been performed. T_{axial} has been shown to be affected by the wind component in the axial

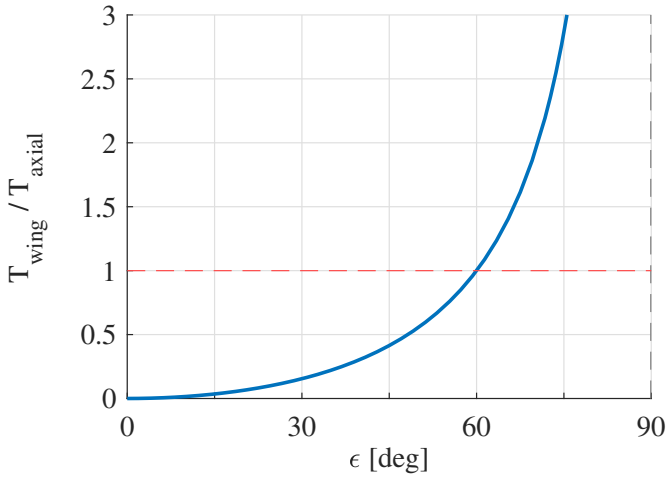


Figure 3.3: Variation of T_{wing}/T_{axial} with ϵ, as determined by Eq. (3.14)

direction and T_{wing} responds to the wind component in the rotor plane direction, in the same manner as a fixed elliptic wing. It will be interesting to analyse the relative influence of each component on the behaviour of the propeller thrust. In order to do that, one approach is to determine the ratio T_{wing}/T_{axial}.

From Eqs. (3.4), (3.8), and (2.29) it is possible to calculate the ratio T_{wing}/T_{axial} to show the contribution of the wing component relative to the axial component on thrust, as follows:

$$\frac{T_{wing}}{T_{axial}} = \frac{2\rho\, V \sin\alpha_p\, w\, S_{disk}\sqrt{\frac{1-\cos\epsilon}{1+\cos\epsilon}}}{2\rho\,(V\cos\alpha_p + w)\, S_{disk}\, w} = \tan\epsilon\sqrt{\frac{1-\cos\epsilon}{1+\cos\epsilon}} \tag{3.13}$$

which with further simplification will result in:

$$\frac{T_{wing}}{T_{axial}} = \frac{(1-\cos\epsilon)}{\cos\epsilon} \tag{3.14}$$

Figure 3.3 depicts the ratio T_{wing}/T_{axial} as a function of the angle ϵ, according to Eq. (3.14). It is clear that the influence of T_{wing} grows with ϵ. At $\epsilon = 60^o$, T_{wing} equalizes with T_{axial}, to largely overcome it at higher angles.

Having determined at which angle the influence of T_{wing} overcomes T_{axial}, we now proceed to relate the T_{wing}/T_{axial} ratio to more convenient inputs, such as AoA and

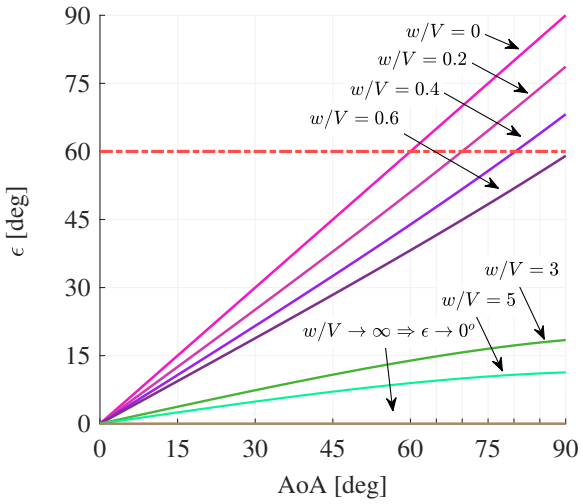


Figure 3.4: Variation of ϵ with AoA, for several w/V values, as determined by Eq. (2.29)

w/V. In order to do that, initially Eq. (2.29) is utilized to relate ϵ to AoA and w/V, in the form $\epsilon = \arctan\left(\frac{\sin\alpha_p}{\cos\alpha_p + w/V}\right)$.

Figure 3.4, plots the relation of ϵ with AoA, for w/V values ranging from 0 to ∞. The lower slope curves represent higher values of w/V. For $w/V \to \infty$, $\epsilon \to 0^o$, which corresponds to hover or static operation condition. Also, for axial conditions, $AoA = 0^o$, ϵ will always be zero for all w/V situations, where the two angles coincide at the origin of the graph. For $w/V = 0$, ϵ always equals AoA, which means $\alpha_{slp} = 0^o$ when the propeller cannot turn the flow, as the wind speed is very high relative to w. It is also of particular interest to evaluate the range at which $\epsilon \geq 60^o$, where T_{wing} overcomes T_{axial}, as per Fig. 3.3. Highlighted in the graph are w/V values from 0 to 0.6, which represent the conditions at which, ϵ can attain values higher than 60^o. It can be seen that at $AoA = 60^o$ the only situation where $\epsilon = 60^o$, is for $w/V = 0$. As AoA values grow higher than 60^o, not so much speed is necessary for T_{wing} to overcome T_{axial}. For instance, at $AoA \approx 70^o$, ϵ will be higher than 60^o for w/V values lower than 0.2. At $AoA \approx 80^o$, $w/V < 0.4$ will provide $\epsilon > 60^o$. At $AoA = 90^o$, values of w/V lower than ≈ 0.6 will be able to produce $\epsilon > 60^o$. For $AoA < 60^o$ or $w/V > 0.6$, ϵ can never reach 60^o. In those conditions, T_{wing} can never overcome T_{axial}, according to Fig. 3.3.

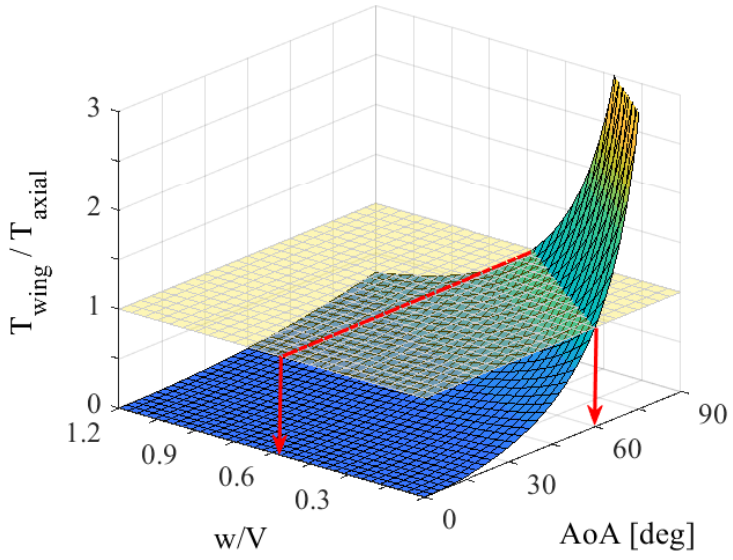


Figure 3.5: The ratio T_{wing}/T_{axial} as a function of AoA and w/V as determined by Eq. (3.15). The region where T_{wing} overcomes T_{axial} is highlighted

Now, in order to express T_{wing}/T_{axial} as function of AoA and w/V, we substitute Eqs. (2.29), (3.10), and (3.11) into Eq. (3.13) to obtain:

$$\frac{T_{wing}}{T_{axial}} = \frac{(\sin\alpha_p)^2}{\left[\cos\alpha_p + \left(\frac{w}{V}\right)\right]\left[\sqrt{1+2\left(\frac{w}{V}\right)\cos\alpha_p + \left(\frac{w}{V}\right)^2} + \cos\alpha_p + \left(\frac{w}{V}\right)\right]} \tag{3.15}$$

Figure 3.5 illustrates the variation of T_{wing}/T_{axial} with AoA and w/V, determined by Eqs.(3.15). The region where $T_{wing} \geq T_{axial}$ is highlighted. It can be seen that the contribution of T_{wing} to the total thrust $T_{wing}+T_{axial}$ increases with AoA and with V. Again, one notices that as w/V is reduced to lower than 0.6 and AoA grows higher than 60^o, eventually T_{wing} overcomes T_{axial} to become dominant, i.e. $T_{wing}/T_{axial} > 1$. At high AoA and at high speeds ($w/V \to 0$), then $T_{wing}/T_{axial} \to \infty$, and T is composed mainly of T_{wing}. In those cases, $T \to 2\rho V w S_{disk}$. At $V = 0$ in hover or in static condition, it becomes $T = 2\rho\, S_{disk}\, w^2$. This agrees well with the findings reported by Glauert, [31, p.319].

To find the exact conditions where T_{wing} equalizes T_{axial}, it is possible to solve Eq. (3.15), setting the ratio at 1. For instance, in the extreme conditions highlighted

in red in Fig. 3.5, when $w/V \to 0$, the solution can be found to be at $AoA = 60^o$, and for the other extreme, when $AoA = 90^o$, $w/V = 0.57735$.

It is possible to rewrite Eq. (3.15), as $T = T_{axial}(1 + T_{wing}/T_{axial})$ to express thrust for all angles of incidence as a function of T_{axial}, AoA and w/V, as follows:

$$T = T_{axial}\left[1 + \frac{(\sin\alpha_p)^2}{\left[\cos\alpha_p + \left(\frac{w}{V}\right)\right]\left[\sqrt{1 + 2\left(\frac{w}{V}\right)\cos\alpha_p + \left(\frac{w}{V}\right)^2} + \cos\alpha_p + \left(\frac{w}{V}\right)\right]}\right] \tag{3.16}$$

3.5 Relating V_{ult} to V_{disk} at Incidence

In order to establish a comparison with axial momentum theory, we wish to obtain relations among V, V_{disk} and V_{ult}, so we recall again the energy balance, and use Eqs. (2.31) and (2.32) one more time:

$$P_{total} = TV_{disk}\cos\epsilon = \frac{1}{2}\dot{m}(V_{ult}^2 - V^2) \tag{3.17}$$

Recalling Eq. (2.9) for thrust, leads to:

$$(V_{ult}\cos\epsilon_{ult} - V\cos\alpha_p)V_{disk}\cos\epsilon = \frac{1}{2}(V_{ult}^2 - V^2) \tag{3.18}$$

which leads to the relation between V_{disk}, V_{ult} and V for propellers at incidence to be:

$$V_{disk} = \frac{1}{2\cos\epsilon}\left[\frac{(V_{ult} - V)(V_{ult} + V)}{V_{ult}\cos\epsilon_{ult} - V\cos\alpha_p}\right] \tag{3.19}$$

For the case of no incidence, $AoA = \epsilon = \epsilon_{ult} = 0^o$, then Eq. (3.19) takes the form of the usual relation from axial momentum theory analysis:

$$V_{disk} = \frac{V_{ult} + V}{2} \tag{3.20}$$

For the limit analysis of angles approaching 90^o, we will use the trigonometric relationships from Fig. 2.2. First, for $AoA \to 90^o$, and for any $\epsilon \neq 90^o$, the scheme

of velocities vectors at the disk and at the ultimate section of the streamtube will be according to the following Fig. 3.6, which clearly allows us to write the relationships in Eqs. (3.21).

Figure 3.6: Scheme of velocities vectors for rotor disk at $AoA = 90^o$

$$V_{disk}^2 = V^2 + w^2 \tag{3.21a}$$

$$V_{ult}^2 = V^2 + (2w)^2 \tag{3.21b}$$

leading, after a simple algebraic manipulation, to:

$$V_{disk} = \frac{1}{2}\sqrt{V_{ult}^2 + 3V^2} \tag{3.22}$$

Now, for $\epsilon \to 90^o$, which can only occur if $AoA = 90^o$, and theoretically at very high translational speeds V, or for $w/V \to 0$, it can be seen from Fig. 3.6 that also $\epsilon_{ult} \to 90^o$, and then $V_{disk} \to V$ and $V_{ult} \to V$, which implies that $V_{ult} \approx V_{disk} \approx V$.

3.6 Power and Efficiency Under the Scope of the New Development

In Section 2.2.4 of Chapter 2, power and ideal efficiency were analysed according to momentum theory, where losses are neglected. When blade profile aerodynamic drag is taken into account, one must consider the power to produce the necessary torque associated with the propeller rotation. The efficiency of the real rotor can then be defined as the ratio of the thrust power, over the torque power, where thrust power is the useful output power according to Eq. (2.34), and the input torque power defined

by the product of the torque times the angular velocity, $Q\Omega$. Thus, the efficiency η of the propeller at incidence can be written as:

$$\eta = \frac{TV\cos\alpha_p}{Q\Omega} = \frac{TV\cos\alpha_p}{Q\,2\pi\,n} \tag{3.23}$$

which can be expressed in terms of thrust and torque coefficients, and the advance ratio, as follows:

$$\eta = \frac{C_T}{2\pi\,C_Q}\,J\cos\alpha_p \tag{3.24}$$

The useful power in Eq. (2.34) can be expressed also in terms of the contribution of each thrust component:

$$P_{useful} = T_{axial}\,V\cos\alpha_p + T_{wing}\,V\cos\alpha_p \tag{3.25}$$

which allows the efficiency to be expressed in terms of the contribution of each component, as:

$$\eta = \eta_{axial} + \eta_{wing} = \frac{C_{Taxial}}{2\pi\,C_Q}\,J\cos\alpha_p + \frac{C_{Twing}}{2\pi\,C_Q}\,J\cos\alpha_p \tag{3.26}$$

3.7 Concluding Remarks

In this chapter, a new theoretical development of the Classical Momentum Theory, applied for angles of incidence, is shown. The main feature of the theory is to state that thrust can be divided into two components. This new theoretical methodology will next be investigated experimentally, through wind tunnel tests, presented in the following chapters.

Chapter 4

Experimental Wind Tunnel Methods

4.1 General Tests Setup and Procedures

In order to investigate the new theoretical development, experimental tests were prepared for the analysis of propellers behaviour, at angles of incidence from 0 to 90^o. The experimental tests were conducted at the University of Canterbury's closed circuit subsonic wind tunnel that has a rectangular cross section of 0.9 m in height by 1.20 m in width, providing maximum test speeds of 60 m/s. The tests conditions were at Reynolds numbers $Re > 18\text{x}10^5$, considering the test section hydraulic diameter of approximately 1 m. Gauze screens are used to reduce non-uniformities in velocity across the flow, and turbulence intensity. Maximum velocity profile variations across the working section were determined to be approximately 0.25%. A complete description of the wind tunnel architecture and its specifications is available in [79]. The tunnel velocity is set through a LabVIEW program, and measured by a Pitot tube that uses a pressure transducer accurate within ±2.5%. It's readings were verified against a vane anemometer. Two tests campaigns were performed, with two different test rigs assembled. The general setup for the tests are described in this section, whereas, particular details of each campaign are further discussed in the pertinent chapters, 5, and 6.

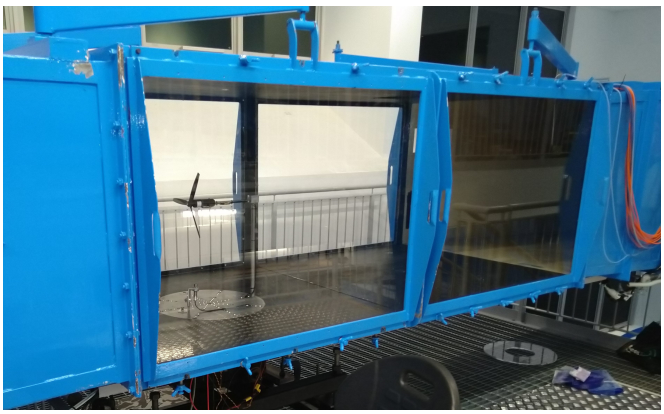
The wind tunnel operates with a 6-axis JR3 45E15A4 force-balance sensor [80], capable of measuring forces and moments in the 3 axis xyz. It can stand loads of

up to 400 N, with nominal accuracy within ±0.25 %. It is installed underneath the test section center on top of a turntable assembly. Load cell readings were verified against fixed weights. A fourth order Butterworth analogue low pass filter with eight different cut-off frequencies, ranging from 6.3 Hz to 926 Hz, can be selected via jumper plugs. The default frequency at the wind tunnel is set to 6.3 Hz for maximum noise attenuation. The force measurements, captured by the wind tunnel force balance, are input to a LabVIEW [81] program that includes another digital low pass filter with adjustable cut-off frequencies. Oscillatory nature of thrust, moments and lateral forces from propellers at incidence are mitigated by choosing a low 1 Hz cut-off frequency. This low frequency has been chosen throughout the tests, in order to allow more steadier average readings, as the aim is to measure static loads. Higher frequencies were tried before the chosen 1 Hz, whereas the average readings obtained in the 1 Hz tests did not differ significantly, from the higher frequency ones, and so it is believed that the low frequency did not cut important information.

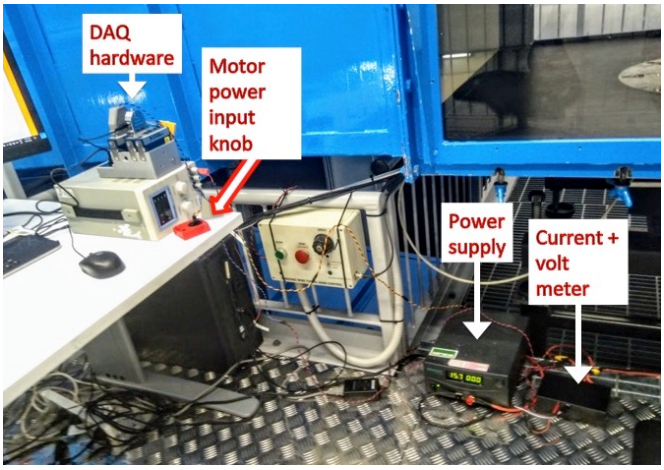
A power supply continuously provides 15.7 V, where the electric current is set according to power requirements to maintain a desired RPM, while complying with maximum currents dictated by limitations of the motors. The power input on the motor is manually controlled through a knob, in order to reach the approximate desired RPM. The motor input current is measured through a current meter that communicates with Data Acquisition hardware (DAQs) from National Instruments, connected to a desktop PC prepared with another LabVIEW program, which records electric power, current, voltage, and motor RPM. An exponential smoothing filter, for RPM readings, was applied in the LabVIEW program. The sensor, used to measure rotational speed of the propeller, is a Monarch [82] remote optical LED sensor, able to measure up to 250,000 RPM from a distance of up to 0.9 m, at a maximum angle of 45^{o}. It is installed at the wind tunnel test section facing the motor, which is prepared with a reflective tape on top of half of its circumference.

Figure 4.1 presents the view of (a) the wind tunnel test section floor, (b) the test section, and (c) the arrangement of the hardware utilized. Figure 4.2 shows (a) the view of the force-balance sensor underneath the wind tunnel test section floor,

(a) External view of the wind tunnel test section floor

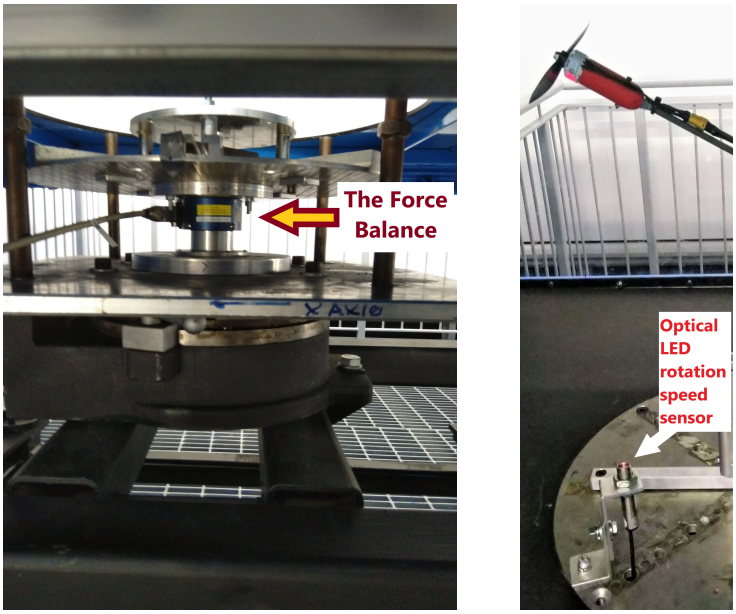


(b) The wind tunnel test section



(c) Hardware utilized for gathering test data at the wind tunnel

Figure 4.1: Closed circuit subsonic wind tunnel at University of Canterbury



(a) (b)

Figure 4.2: Sensors: (a) The force-balance underneath the test-section, (b) The optical LED rotation speed sensor

and (b) the optical LED rotation speed sensor. Figure 4.3 depicts the scheme of the general wind-tunnel assembly with side elevation view on the left, and top view on the right, adapted from [79]. The wind flow is counter-clockwise as seen from the side elevation view. This plan is from the original installation of the 1960s at a different building. The current wind tunnel assembly, at the new recently inaugurated Mechanical Engineering building, features a different two storey floor plan configuration for the test section access, as seen in Fig. 4.1a. The essential features of the wind tunnel, represented as letters A through G in the drawings of Fig. 4.3, are presented in Table 4.1 below:

A	The test section of the wind tunnel
B, C, E	Diffusers to provide pressure recovery, thereby reducing power requirements of the tunnel
D	Fan and fan drive to overcome losses in the wind tunnel circuit
C, F	Corner vanes to reduce losses in turning the flow
G	Settling chamber and contraction to achieve a uniform flow and turbulence reduction, where gauze screens are also installed to achieve similar enhanced effects

Table 4.1: Essential features of the wind tunnel at the University of Canterbury

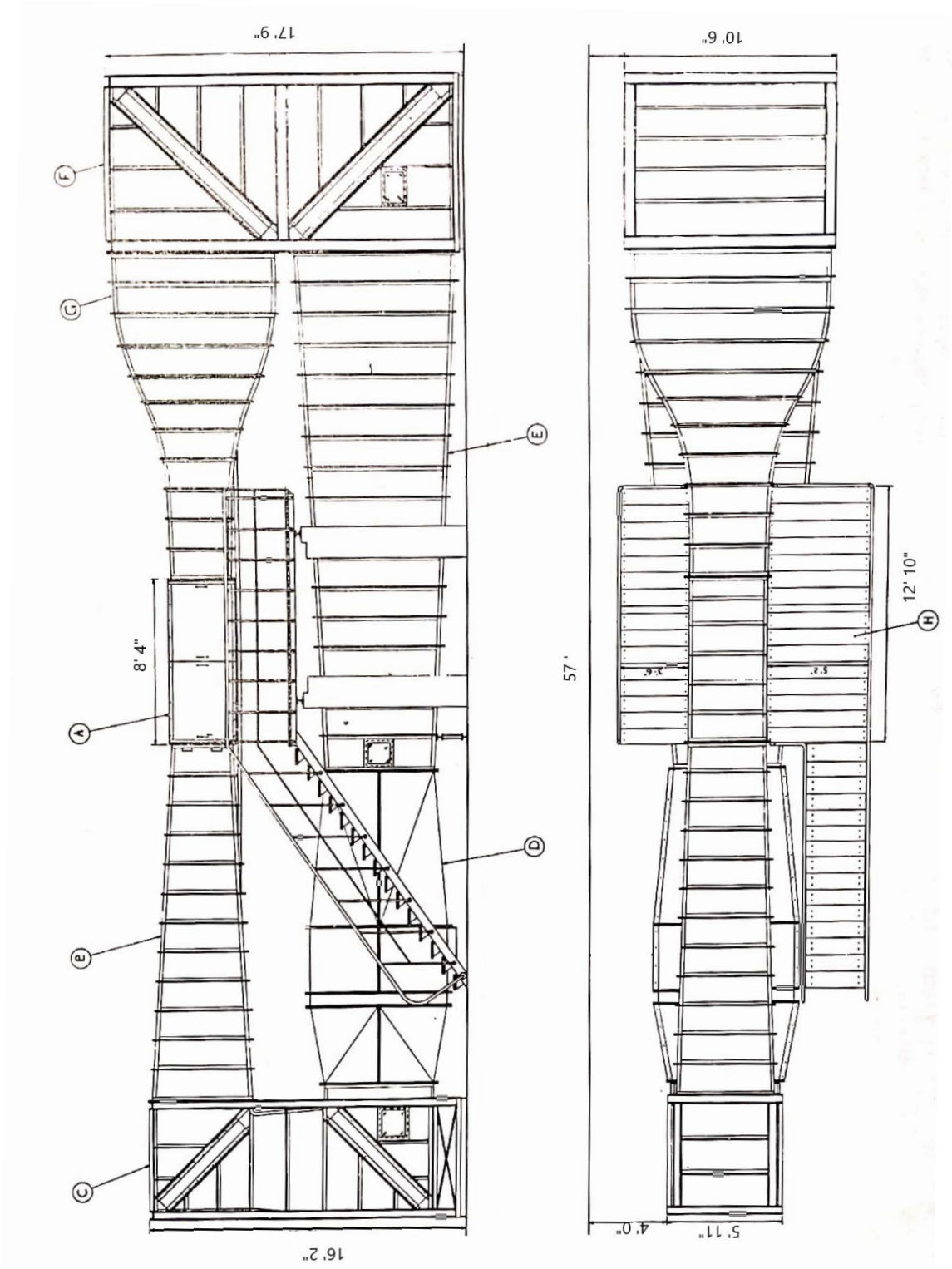
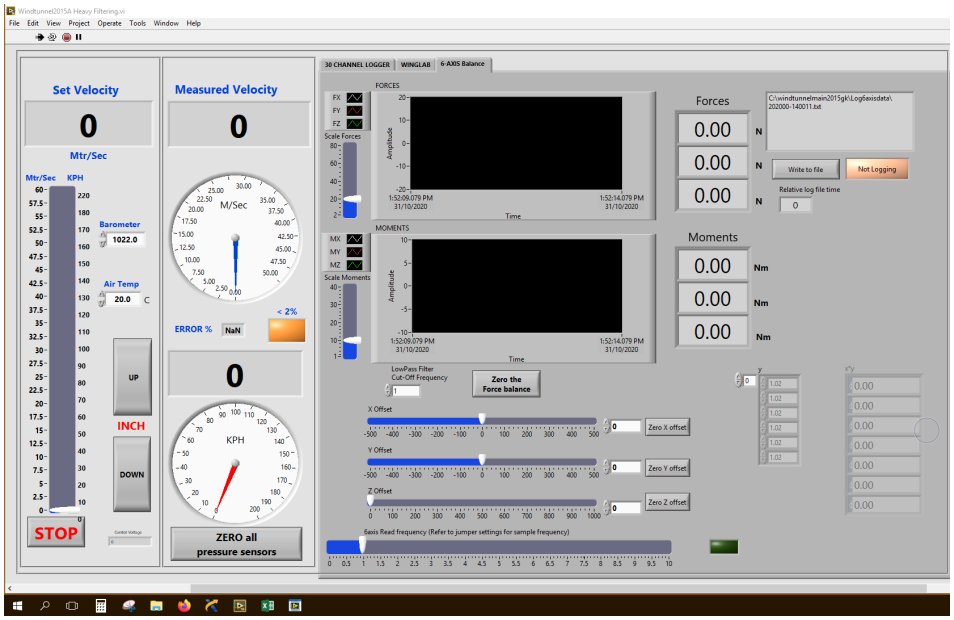
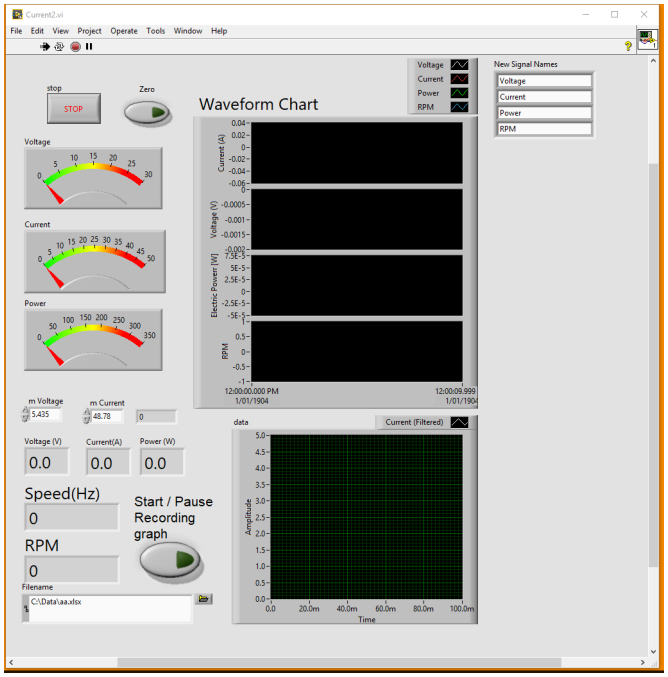


Figure 4.3: Adapted scheme of the wind tunnel at the University of Canterbury. Wind flow is counter-clockwise on the side elevation view, on the left.
Source: [79]

Figure 4.4 presents the screenshots of the LabVIEW programs utilized in the wind tunnel.



(a) Program that sets the wind tunnel velocity and records forces data



(b) Program that records RPM, voltage, current and electric power data

Figure 4.4: Screenshots of the LabVIEW programs

4.2 Data Processing Methods

The main objective of the research is to evaluate the thrust performance at angles of incidence according to the new development, therefore, although normal forces and moments are recorded, the focus of the experimental tests results are on thrust analysis under the scope of the two components of thrust, T_{axial}, and T_{wing}.

Prior to the tests, the holding set without the propeller is exposed to the same wind speeds and AoA values as of the propeller tests, in order to evaluate the aerodynamic resistance of the rig. The forces readings acquired are fitted to spline surface functions. These are netted from the final propeller tests readings. Figure B.2, in Appendix B, displays the relevant forces and moments measured for the rig resistance calculations of the 2nd tests campaign. The extra effect of the propeller slipstream on the holding sting was not evaluated and the total net thrust should be expected to be somewhat higher, especially for larger diameter propellers, where the slipstream cross section will impinge on larger portions of the rig. However, as AoA and the velocity V increase, the slipstream is deflected and should not reach the back rod, affecting only a part of the adjacent motor holding bar. In those conditions, the slipstream effect should be expected to affect mostly the propeller normal force N_p, which acts in the rotor plane. As the main interest of this research is on thrust analysis, the expected error from this effect is not an initial major concern. The effect of the slipstream on thrust should be expected to be larger at low wind tunnel speeds, and low AoA, therefore the actual thrust on those conditions should be expected to be higher than the measurements. A quantitative evaluation of the slipstream influence on the holding rig has not been done due to the limited time for conducting the present research, where new rig and sensors preparation would be deemed necessary. Ideally, load sensors, and a torque transducer should be placed right aft of the propeller.

Wind tunnel corrections for equivalent free airspeed were not applied in the tests and should be negligible for small diameter propellers, although according to Glauert's corrections for propellers in axial tests in closed wind tunnels [83, p.225], some correction of up to 5%-10% should be necessary for propellers larger than 15", for the conditions tested. However, as one of the main objectives of the tests is to evaluate

the relative performance of the two components of thrust, it is assumed that the neglect of the corrections to be non-critical, for the current investigation. The existence of this error, nevertheless, must be acknowledged for large propellers.

Every propeller test condition for AoA, V, and RPM is run for around 20 seconds. The data results for all 6 axis forces are recorded in a .txt file. Three batches of tests are run for every condition, comprising around 80 to 100 measurements per condition. For every test condition, a file is created. These files are stored in a specific folder named after the specific propeller name. The power and RPM data, for every condition, are recorded in an excel file. These excel files are stored in the same folder as the .txt files.

All experimental test data are then compiled and analysed with MATLAB [84]. The macro structure of the tests procedures, and data analysis are presented in a summarized way, through the flowchart depicted in Fig. 4.5. A detailed description of these procedures can be found in Appendix B. The performance calculations are done via object-oriented programming, where a class "propeller.m" was created with several properties (variables of interest), and methods (functions) required for the propeller analysis. For every different propeller, an object of the class will be created. A main script, written to receive the tests compiled data, also generates an object of the class "propeller", and records the returned results in appropriate files. This main script, as well as the code for the class "propeller.m" can be found in Appendix C.

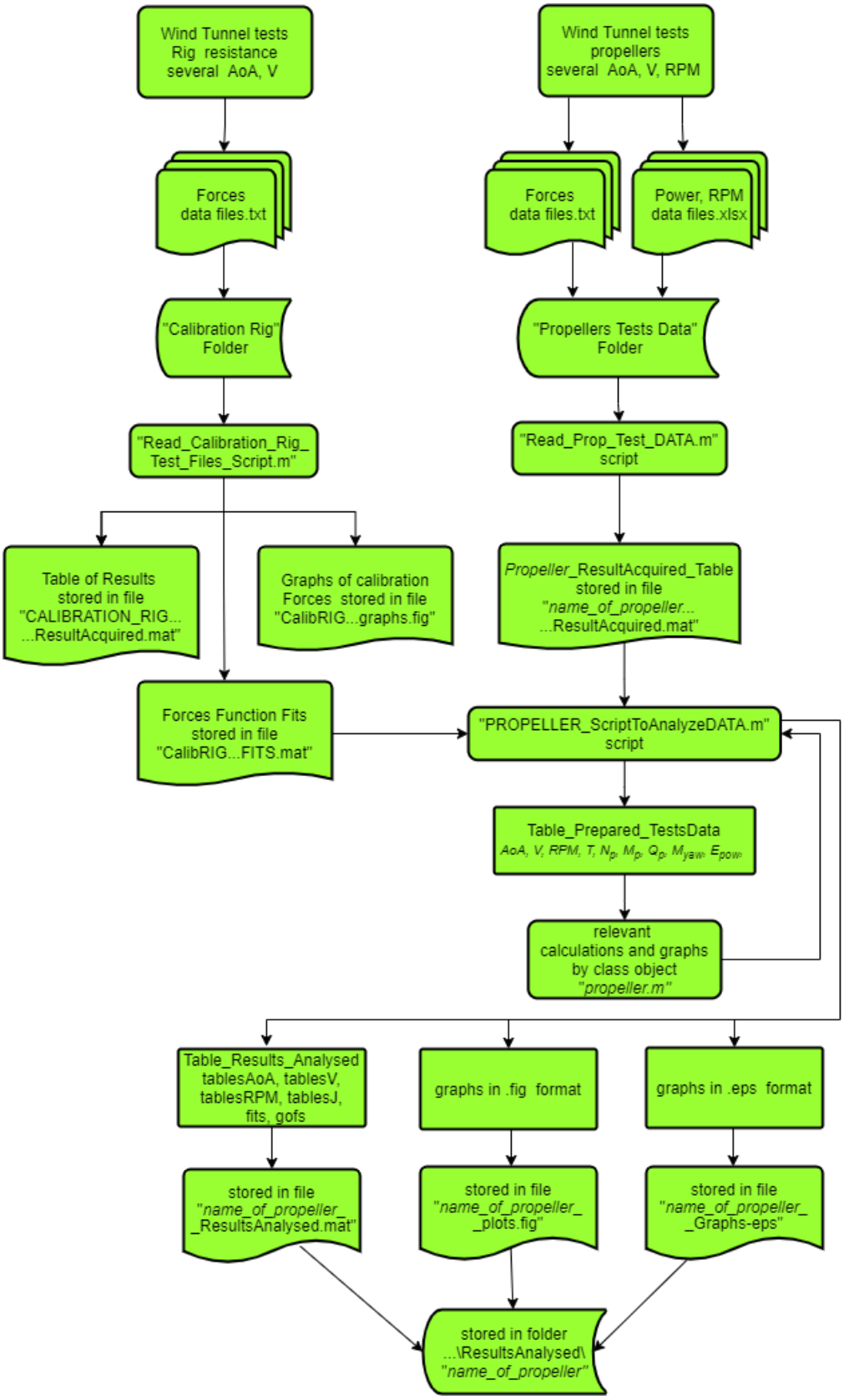


Figure 4.5: Simplified flowchart of the tests procedures and analysis

4.3 Uncertainty Analysis and Error Propagation

For every batch of measurements, the readings were averaged and the precision errors evaluated as one standard deviation, and presented in percentage terms relative to the average value obtained. In some cases, when average forces measured are very low, approaching zero, the errors are evaluated in absolute terms as it is not reasonable to display percentage errors meaninglessly very high. When displayed as percentage errors, the points where the average values are very low are omitted.

The errors propagation rule, based on Taylor expansion, and described in [85], has been utilized to estimate precision errors of the variables of interest, where it is assumed that all variables measurements are uncorrelated, and follow Gaussian distributions. It states that, for a function $f = f(x, y, z...)$, the variance of f, σ_f^2, is:

$$\sigma_f^2 = \left[\left(\frac{\partial f}{\partial x} \sigma_x \right)^2 + \left(\frac{\partial f}{\partial y} \sigma_y \right)^2 + \left(\frac{\partial f}{\partial z} \sigma_z \right)^2 + ... \right] \tag{4.1}$$

Error estimation for J:

From error propagation, the variance of $J = \frac{V}{nD}$ can be written as:

$$\sigma_J^2 = \left(\frac{\partial J}{\partial V} \right)^2 \sigma_V^2 + \left(\frac{\partial J}{\partial n} \right)^2 \sigma_n^2 \tag{4.2}$$

which leads to :

$$\sigma_J^2 = \left(\frac{J}{V} \right)^2 \sigma_V^2 + \left(\frac{-J}{n} \right)^2 \sigma_n^2 \tag{4.3}$$

As $n = RPM/60$, then: $\sigma_n^2 = \sigma_{RPM}^2/3600$. Substituting both in Eq. (4.3) results in:

$$\sigma_J^2 = \left(\frac{J}{V} \right)^2 \sigma_V^2 + \left(\frac{J}{RPM} \right)^2 \sigma_{RPM}^2 \tag{4.4}$$

which further simplifies to the percentage error of J as function of the sum of the percentage errors squared of V and RPM:

$$\left(\frac{\sigma_J}{J}\right) = \sqrt{\left(\frac{\sigma_V}{V}\right)^2 + \left(\frac{\sigma_{RPM}}{RPM}\right)^2} \tag{4.5}$$

Error estimation for C_T:

From error propagation, the variance of $C_T = \frac{T}{\rho\, n^2 D^4}$ can be written as:

$$\sigma_{C_T}^2 = \left(\frac{\partial C_T}{\partial T}\right)^2 \sigma_T^2 + \left(\frac{\partial C_T}{\partial n}\right)^2 \sigma_n^2 \tag{4.6}$$

here, we are neglecting the error of ρ in the analysis. The evaluation of the derivatives yields:

$$\sigma_{C_T}^2 = \left(\frac{C_T}{T}\right)^2 \sigma_T^2 + \left(\frac{-2C_T}{n}\right)^2 \sigma_n^2 \tag{4.7}$$

which further results in the percentage error estimation of C_T as function of the percentage errors fo T and RPM, as:

$$\left(\frac{\sigma_{C_T}}{C_T}\right) = \sqrt{\left(\frac{\sigma_T}{T}\right)^2 + 4\left(\frac{\sigma_{RPM}}{RPM}\right)^2} \tag{4.8}$$

Error estimation for the induced speed w:

From Eq. (2.21), a function $\mathscr{F} = \mathscr{F}(T, V, w)$ can be defined as:

$$\mathscr{F}(T, V, w) = \left(\frac{T}{2\rho\, S_{disk}}\right)^2 - V^2 w^2 - 2V w^3 \cos\alpha_p - w^4 \tag{4.9}$$

For a given angle α_p, and disregarding measurement errors in ρ, then from Eq. (2.21), one can assume that $w = w(T, V)$, so:

$$\sigma_w^2 = \left(\frac{\partial w}{\partial T}\right)^2 \sigma_T^2 + \left(\frac{\partial w}{\partial V}\right)^2 \sigma_V^2 \tag{4.10}$$

which can be expressed as:

$$\sigma_w^2 = \left(\frac{\partial w}{\partial \mathscr{F}}\frac{\partial \mathscr{F}}{\partial T}\right)^2 \sigma_T^2 + \left(\frac{\partial w}{\partial \mathscr{F}}\frac{\partial \mathscr{F}}{\partial V}\right)^2 \sigma_V^2 \tag{4.11}$$

From Eq. (4.9), we get the partial derivatives:

$$\frac{\partial \mathscr{F}}{\partial w} = -2V^2 w - 6Vw^2 \cos\alpha_p - 4w^3 \tag{4.12a}$$

$$\frac{\partial \mathscr{F}}{\partial T} = \frac{2T}{(2\rho\, S_{disk})^2} \tag{4.12b}$$

$$\frac{\partial \mathscr{F}}{\partial V} = -2Vw^2 - 2w^3 \cos\alpha_p \tag{4.12c}$$

Substituting Eqs. (4.12) into Eq. (4.11) leads to:

$$\sigma_w^2 = \left(\frac{2T/(2\rho\, S_{disk})^2}{-2V^2 w - 6Vw^2\cos\alpha_p - 4w^3}\right)^2 \sigma_T^2 + \left(\frac{-2Vw^2 - 2w^3\cos\alpha_p}{-2V^2 w - 6Vw^2\cos\alpha_p - 4w^3}\right)^2 \sigma_V^2 \tag{4.13}$$

The first parenthesis is now multiplied by T^2, whereas σ_T^2 divided by T^2. Also, by multiplying the second parenthesis by V^2, while σ_V^2 is divided by V^2, will result in:

$$\sigma_w^2 = \left(-\frac{T^2/(2\rho\, S_{disk})^2}{V^2 w + 3Vw^2\cos\alpha_p + 2w^3}\right)^2 \left(\frac{\sigma_T}{T}\right)^2 + \left(\frac{V^2 w^2 + Vw^3\cos\alpha_p}{V^2 w + 3Vw^2\cos\alpha_p + 2w^3}\right)^2 \left(\frac{\sigma_V}{V}\right)^2 \tag{4.14}$$

Now, substituting $T^2/(2\rho\, S_{disk})^2$ for the rhs of Eq. (2.21), and dividing all by w, results in:

$$\left(\frac{\sigma_w}{w}\right) =$$

$$\sqrt{\left(\frac{V^2 w + 2Vw^2\cos\alpha_p + w^3}{V^2 w + 3Vw^2\cos\alpha_p + 2w^3}\right)^2 \left(\frac{\sigma_T}{T}\right)^2 + \left(\frac{V^2 w + Vw^2\cos\alpha_p}{V^2 w + 3Vw^2\cos\alpha_p + 2w^3}\right)^2 \left(\frac{\sigma_V}{V}\right)^2} \tag{4.15}$$

which is the expression for the percentage error in w, as function of the percentage errors in T, and V measured.

The estimation of thrust errors depends on the rig geometry, so it will be described further in the appropriate Sections 5.2, and 6.3, where the first and second round of tests with their different rigs are presented.

Chapter 5

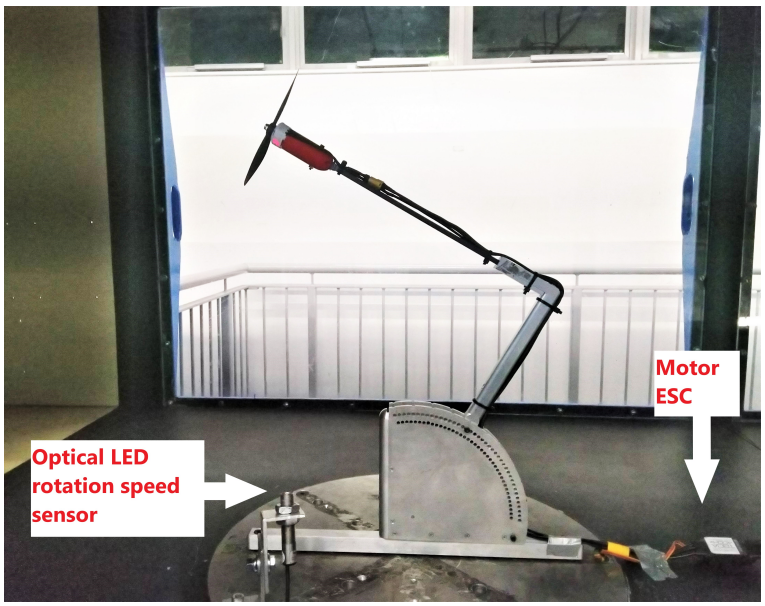
Initial Wind Tunnel Tests

5.1 1st Campaign Test Rig Configuration and Setup

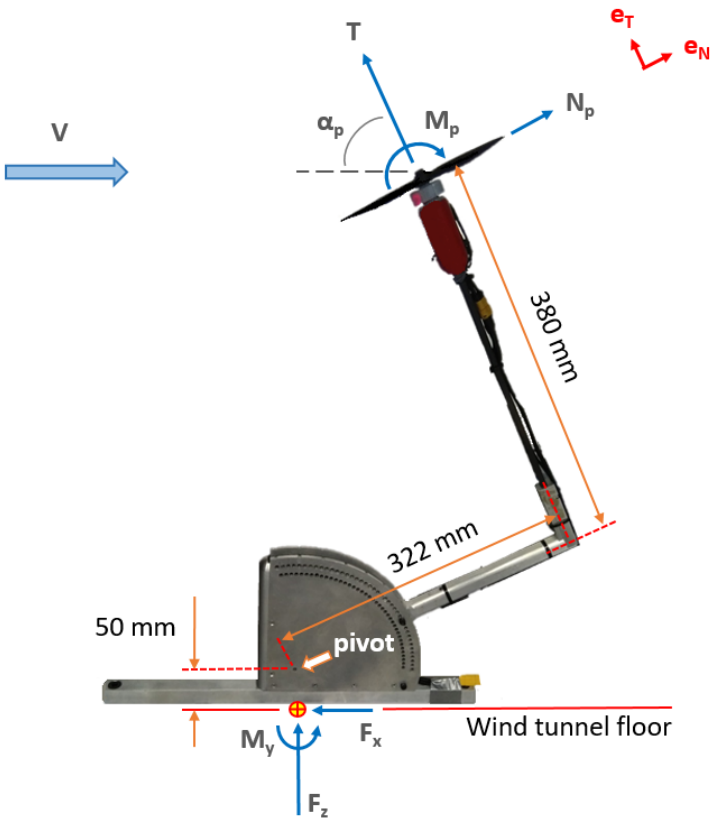
Due to a temporary impossibility to utilize the wind tunnel turntable assembly, which allows for variations in the angles of incidence, an aluminum rig, with variable *AoA* settings, was manufactured for the initial round of tests. The rig setup allows *AoA* settings from 0^o to 90^o, with minimum adjustments of 2.5^o. The aluminium rig holds a 6 mm carbon fibre square rod and a 3D printed motor/propeller assembly, shown in Fig. 5.1a.

Initially, the tests were conducted using a propeller that was readily available from the mechatronics laboratory stock. The propeller is a 2 blade "HQ-6x4.5", manufactured by HQprop [86]. It has a diameter of 6 inch and a 4.5 inch/revolution fixed pitch, implying a blade pitch angle $\beta_{0.75} \approx 17.6^o$. An Ethix TeamBlacksheep Silk V2 2345V electric motor [87], is utilized and an 80A T-MOTOR [88] electronic speed controller (ESC) used to control the motor. The recommended ESC specification for the motor electric current would be up to 30A, however the current limitation was set through the power supply.

Figure 5.2a shows the front view of the assembly in the wind tunnel. Figure 5.2b displays a close-up of the motor, propeller and 3-D printed motor-holding nacelle, while in Fig. 5.2c, the Electronic Speed Controller (ESC) utilized in the tests is shown.

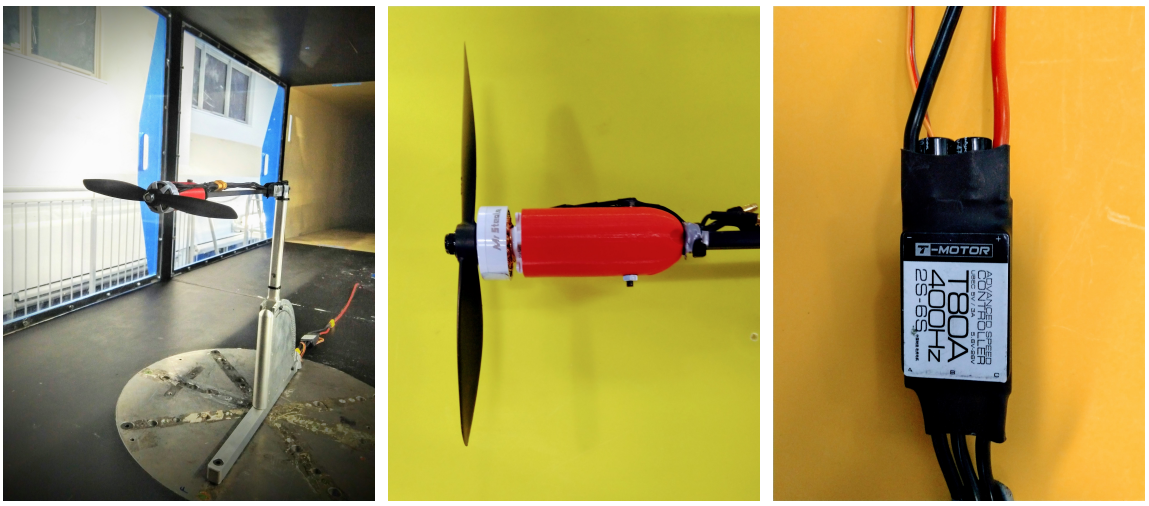


(a) Test rig hardware



(b) Scheme of propeller forces for the wind tunnel tests

Figure 5.1: Rig utilized in the 1st tests campaign



(a) Front view of propeller HQ-6x4.5 test assembly

(b) The propeller, motor and 3D printed nacelle assembly

(c) The electronic speed controller used in the tests

Figure 5.2: The motor, propeller, and ESC of the 1st tests campaign

For wind speeds of 25 m/s and AoA higher than 60° it was not possible to reach rotations higher than 15,000 rpm as power requirement exceeded motor limitations. Curve fits, when applied for the data analysed, have confidence interval not lower than 95 and with adjusted R^2 correlation coefficient not lower than 0.95.

Propeller thrust and normal forces readings were acquired at wind tunnel speeds ranging from zero to 25 m/s, and motor speed rotations from 9,000 to 18,000 rpm. Angles of attack ranged from 0° to 90°. A total of 182 test conditions were evaluated. Some were outliers and ignored in the results analysis.

Equations (5.1a) and (5.1b) enable the calculation of T and N_p from forces F_x and F_z measured at the wind tunnel tests, according to the scheme depicted in Fig. 5.1b. The analysis of experimental test data is performed by utilizing the thrust values acquired for every test condition, and subsequently inputting those into Eq. (2.21) to calculate the associated induced speeds w. Once w is obtained, then all the remaining calculations can be performed.

$$F_x = \cos\alpha_p \, T - \sin\alpha_p \, N_p \tag{5.1a}$$

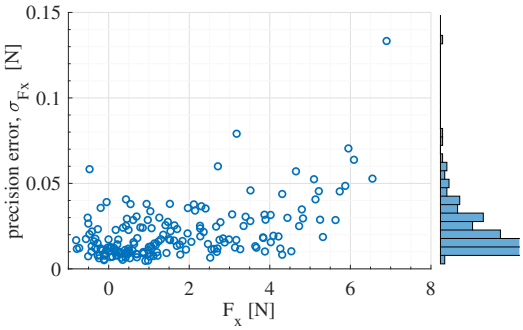
$$F_z = \sin\alpha_p \, T + \cos\alpha_p \, N_p \tag{5.1b}$$

5.2 Evaluation of Results Errors

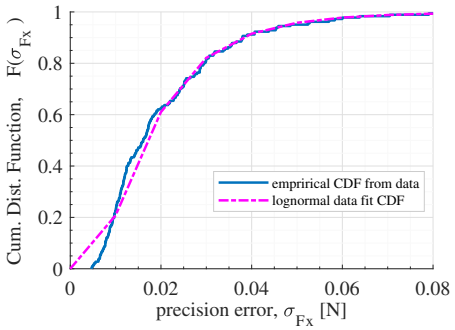
In order to estimate the errors in thrust for the tests, the geometry of the rig will be determinant for the calculations. Thrust is obtained from indirect measurements of F_x, and F_z, and can be expressed, according to Eqs. (5.1a) and (5.1b), as $T = \cos\alpha_p\, F_x + \sin\alpha_p\, F_z$. Then the estimation of the errors σ_T , are inferred from F_x and F_z error propagation, as:

$$\sigma_T = \sqrt{(\cos\alpha_p\, \sigma_{F_x})^2 + (\sin\alpha_p\, \sigma_{F_z})^2} \tag{5.2}$$

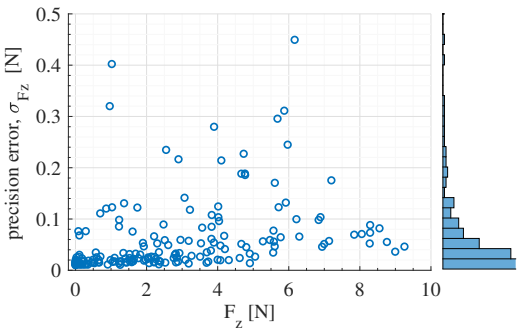
All measurements obtained for F_x and their respective errors σ_{F_x} are presented in Fig. 5.3a, where it is seen that the highest error was 0.14 N. A histogram plot representing, qualitatively, the probability distribution function of σ_{F_x} also appears



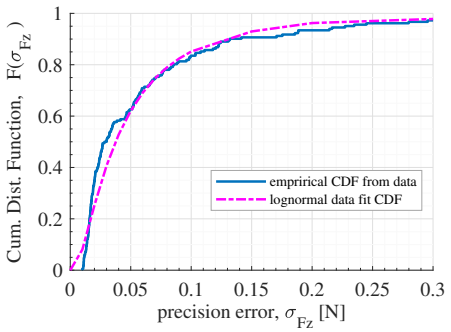
(a) Measurement precision errors in F_x with probability density function



(b) Cumulative distribution function of errors in F_x



(c) Measurement precision errors in F_z with probability density function



(d) Cumulative distribution function of errors in F_z

Figure 5.3: Measurement Precision Errors in F_x, and F_z for propeller HQ6x4.5

and indicates the region of results highest concentration. The following graph, Fig. 5.3b, then presents a quantitative analysis of the distribution. It depicts the empirical data cumulative distribution function, where it can be seen that around 80% of the measurement precision errors lie below $\sigma_{F_x} \leq 0.03$ N. Also a log-normal fit is applied to the data to show that it appears to follow a log-normal distribution pattern. For F_z, the same analysis is performed, where around 80% of the errors are found to be below 0.1 N, as illustrated in Fig. 5.3c and by the the cumulative distribution function of σ_{F_z} in Fig. 5.3d.

Figures 5.4 (a), (c), (e), (g) present the data points acquired for V, RPM, J and T and their respective estimated errors. The cumulative distribution functions for σ_V/V points are presented in Fig. 5.4(b), for σ_{RPM}/RPM in Fig. 5.4(d), for σ_J/J in Fig. 5.4(f), and for σ_T/T in Fig. 5.4(h). The calculation of the errors in J is done through error propagation estimation, according to Eq. (4.5) in subsection 4.3. For the estimation of the error in T, Eq. (5.2) is utilized.

In Fig. 5.5, the same analysis is presented for C_T and the induced velocity w. Highlighted in Figs. 5.4 (b), (d), (f), (h) and Figs. 5.5 (b), (d) are the parameters that represent the mean μ and standard deviation s of the logarithmic values, that characterize the lognormal distribution fit, from MATLAB, see [89].

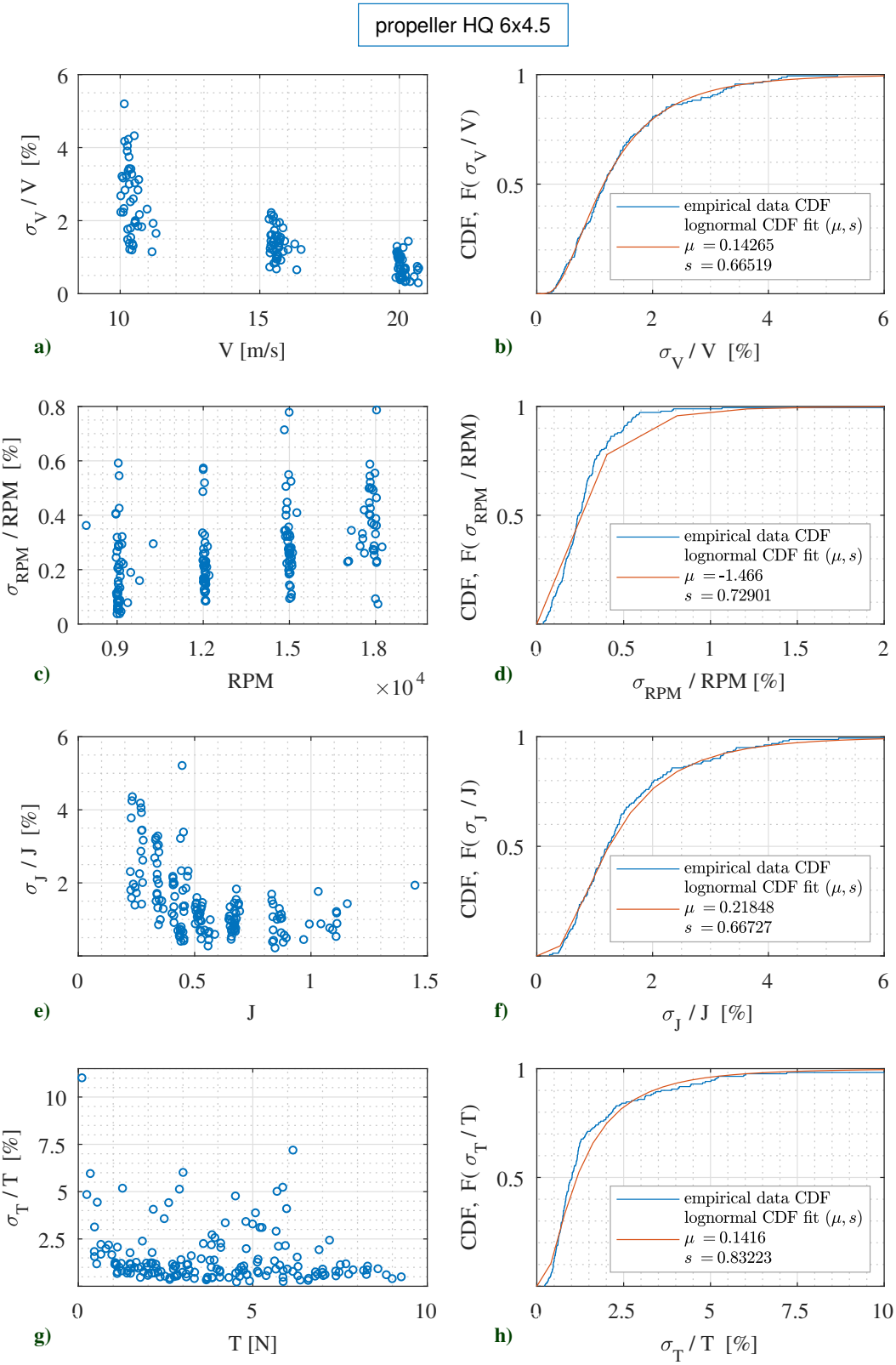


Figure 5.4: Measurement precision errors for propeller HQ6x4.5: (a) in V, (c) in RPM, (e) in J, (g) in T, and (b), (d), (f), (h) are their respective Cumulative Distribution Functions.

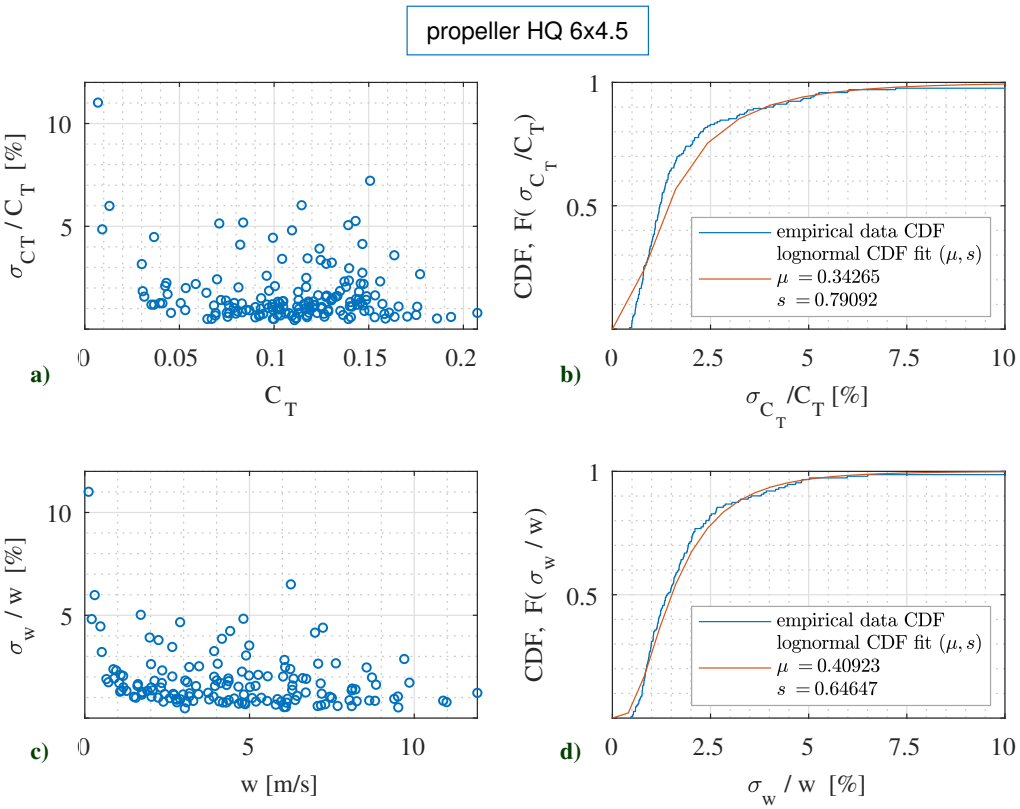


Figure 5.5: Measurement precision errors for propeller HQ6x4.5: (a) in C_T, (c) in w, and (b), (d) are their respective Cumulative Distribution Functions.

5.3 Experimental Results and Discussion

5.3.1 Thrust Measurements Analysis

Thrust analysis, under influence of the three main variables AoA, RPM, V, is presented hereafter.

Thrust performance of the propeller, as a function of RPM, at different free-stream velocities V, at no incidence, ($AoA = 0^o$), is illustrated in Fig. 5.6. It can be seen that, as V increases at constant RPM, (meaning a raise in J), a reduction in the available thrust occurs. This is in agreement with [39, p.78], and in general for propellers tested in [26, 28, 48, 49, 90–93]. In the cases of exceedingly high blade angles $\beta_{0.75} > 50^o$, McLemore & Cannon [26] found a first increase of C_T with J, prior to the following reduction as J continued to grow. Thrust also exhibits a parabolic variation with RPM. These behaviour patterns are also presented in [26, 28, 48, 49, 90–93].

Figure 5.7 illustrates the influence of AoA on thrust for a constant wind speed $V = 20\,m/s$. The thrust T is increased, as the angle of incidence rises. This finding is consistent with the concluding remarks in [26–28, 48, 49].

Figure 5.8 presents the thrust measured from the propeller as function of AoA and RPM, for (a) $V = 10\,m/s$ and (b) $V = 20\,m/s$. Again from both graphs at $AoA = 0^o$, it can be seen that the thrust is reduced with the increase of V. For instance, at 18000 rpm, in Fig. 5.8(a), at $V = 10\,m/s$, $T = 6\,N$, while in Fig. 5.8(b) at $V = 20\,m/s$, at the same RPM, $T = 4\,N$ approximately. As AoA rises, T increases in both cases. However, the slope of thrust increase with AoA, $\partial T/\partial \alpha_p$, is higher in the case of the higher speed. Another interesting observation from Fig. 5.8 is that at low AoA, and low RPM, for both cases that present incoming air speeds, no thrust

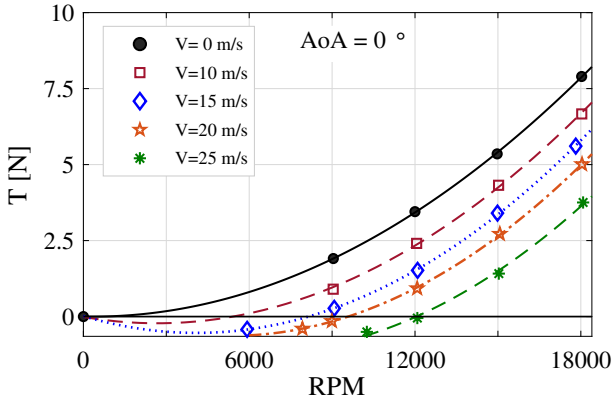


Figure 5.6: Variation of thrust T with RPM, in axial flow ($AoA = 0^o$), as V is set to 5 different values

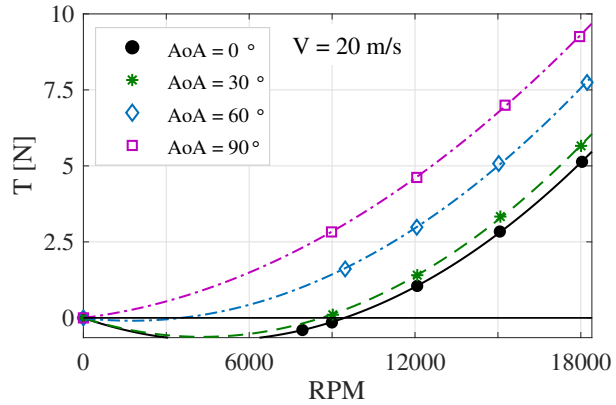


Figure 5.7: Variation of thrust T with RPM, at $V = 20m/s$, for different AoA values.

is available, which is also seen in Figs. 5.6 and 5.7. In that region, the propeller will perform in a windmill state until a sufficient value of RPM is achieved. For example, in Fig. 5.8(a) where $V = 10\,m/s$ no thrust is generated before the propeller reaches around 5000 rpm at low angles of attack. For the case of $V = 20\,m/s$ in Fig. 5.8(b), the value required is 9500 rpm at no incidence and the region of no thrust achievable ($T \leq 0$) is wider, reaching up to $AoA \approx 60^o$. However, as AoA is increased, less RPM is required to achieve some thrust in both cases.

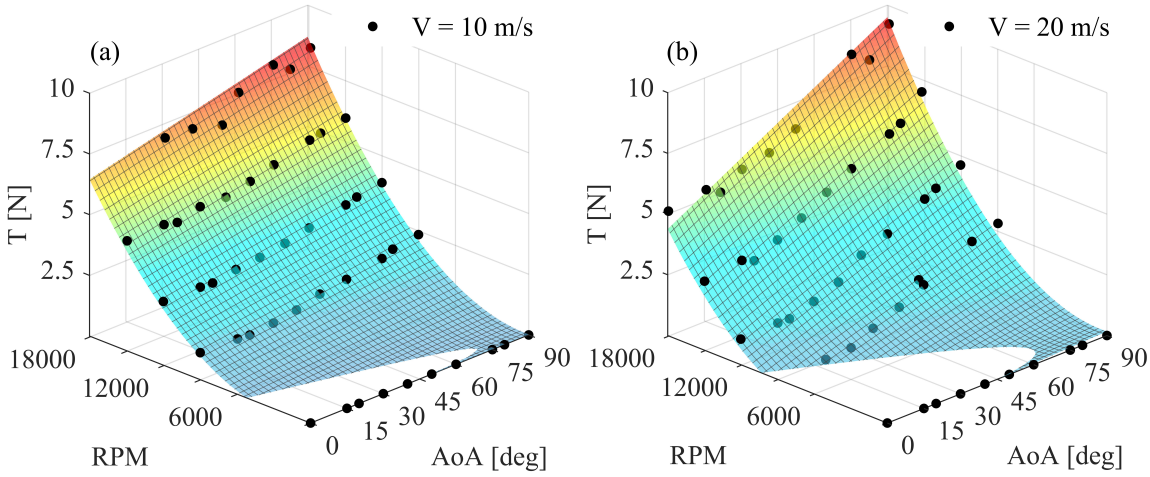


Figure 5.8: Thrust T varied with AoA, and RPM: (a) V=10 m/s, (b) V=20 m/s.

Figure 5.9 shows how thrust T is varied with AoA, for $V = 10$, $V = 15$ and $V = 20\,m/s$ and rotation speeds of 12000, 15000 and 18000 rpm. In all tested cases, the thrust T is found to always increase with AoA, more noticeably for $AoA > 20^o$. A higher slope $\partial T/\partial \alpha_p$ is associated with a higher V, for a given RPM. Therefore, the analysis here means to be extended also to the advance ratio J, as an increase in velocity, at a given constant RPM, represents also an increase in J, and vice-versa for a decrease. This difference in slopes will cause a change in thrust sensitivity to V at high angles of incidence; while at low angles T decreases with V, ($\partial T/\partial V < 0$), for AoA around 60° and over, T eventually starts to increase with increased V, ($\partial T/\partial V > 0$). Similar results for C_T growing with AoA and the eventual inversion of behavior is found in the experimental tests in Refs. [48, p.85, 90], [49].

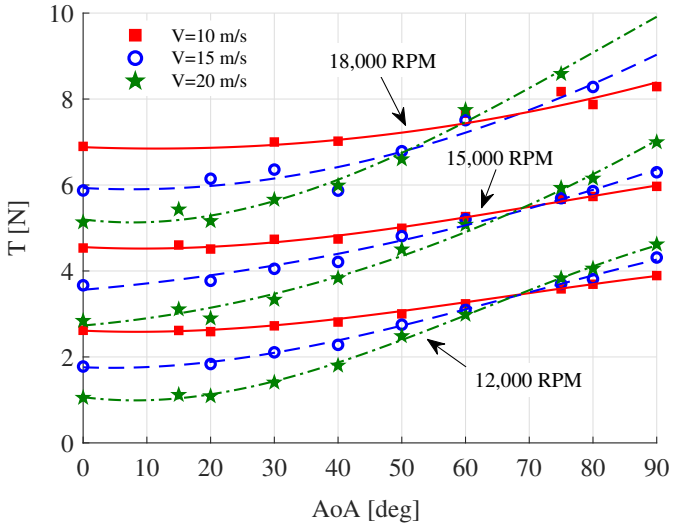


Figure 5.9: Variation of T with AoA at different RPM, as V is set to 3 different values.

Constant AoA surfaces are depicted in Fig. 5.10 that shows T as function of V and RPM. The $AoA = 0^o$ surface, below $AoA \approx 60^o$, T decreases with V, or $(\partial T/\partial V < 0)$, for any given RPM, whereas for $AoA > 60^o$, T grows with V, or $(\partial T/\partial V > 0)$. For $AoA \approx 60^o$ it can be seen that a small sensitivity to V, still negative occurs.

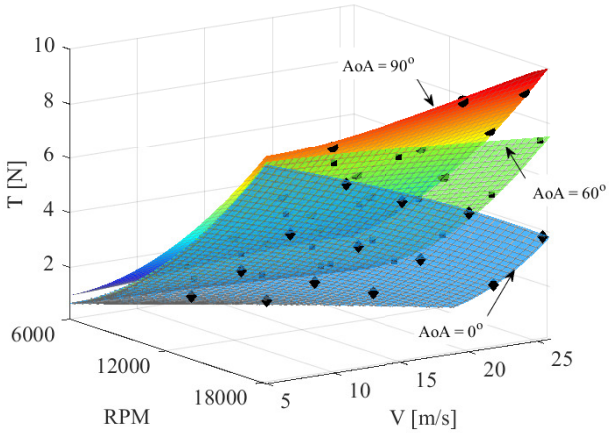


Figure 5.10: T vs V and RPM for constant AoA surfaces. Note the different sensitivities to speed at different AoA surfaces.

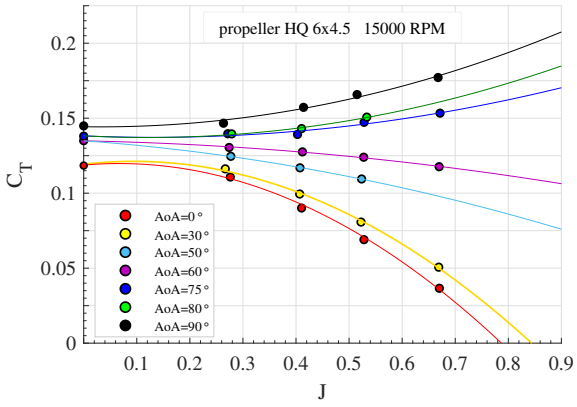


Figure 5.11: Variation of C_T as a function of J for different AoA at 15,000 rpm

Figure 5.11 illustrates C_T against J, for $RPM = 15,000$, which again agrees with the results obtained by Ariza [48, p.85, 90, 142], and Serrano et. al [49] for fixed low pitch propellers, where the inversion of sensitivity $\partial C_T/\partial J$ occurs at $AoA \approx 60^o$ or higher. The results here, obtained at $J = 0$ show that C_T measured increased with increasing AoA. This is probably associated with ground effects from the wind tunnel. The expected behavior would be the same static thrust at all incidence angles.

Figure 5.12(a) shows the thrust T contour surface at 15,000 rpm. The analysis here, at constant RPM, again means that an increase or decrease in V is also associated with a corresponding increase or decrease in J. Again at a low AoA value, T is decreased as V is growing. However, at a high AoA value, beyond 60°, the thrust variation behavior inverts, to increase with V. Also T always increases with AoA at constant speeds. T_{axial} is presented in Fig. 5.12(b), and T_{wing} depicted in Fig. 5.12(c), calculated according to Eqs. (2.20), (3.4) and (3.8). T_{axial} behaves in the same way for all angles of incidence AoA, decreasing as V is increased. T_{wing} is rising with V and AoA, peaking at $AoA = 90^o$ and at higher speeds. The slope of T_{wing} increase with AoA, $(\partial T_{wing}/\partial \alpha_p)$ also grows with V. Note in Fig. 5.12(a) and (b), the slight increase in T with AoA, for V=0 (static thrust tests performed at different AoA), which is the same result shown in 5.11 at $J = 0$, being interpreted as ground effects from the wind tunnel.

Figure 5.13 shows the measured T and its components T_{axial} and T_{wing} calculated

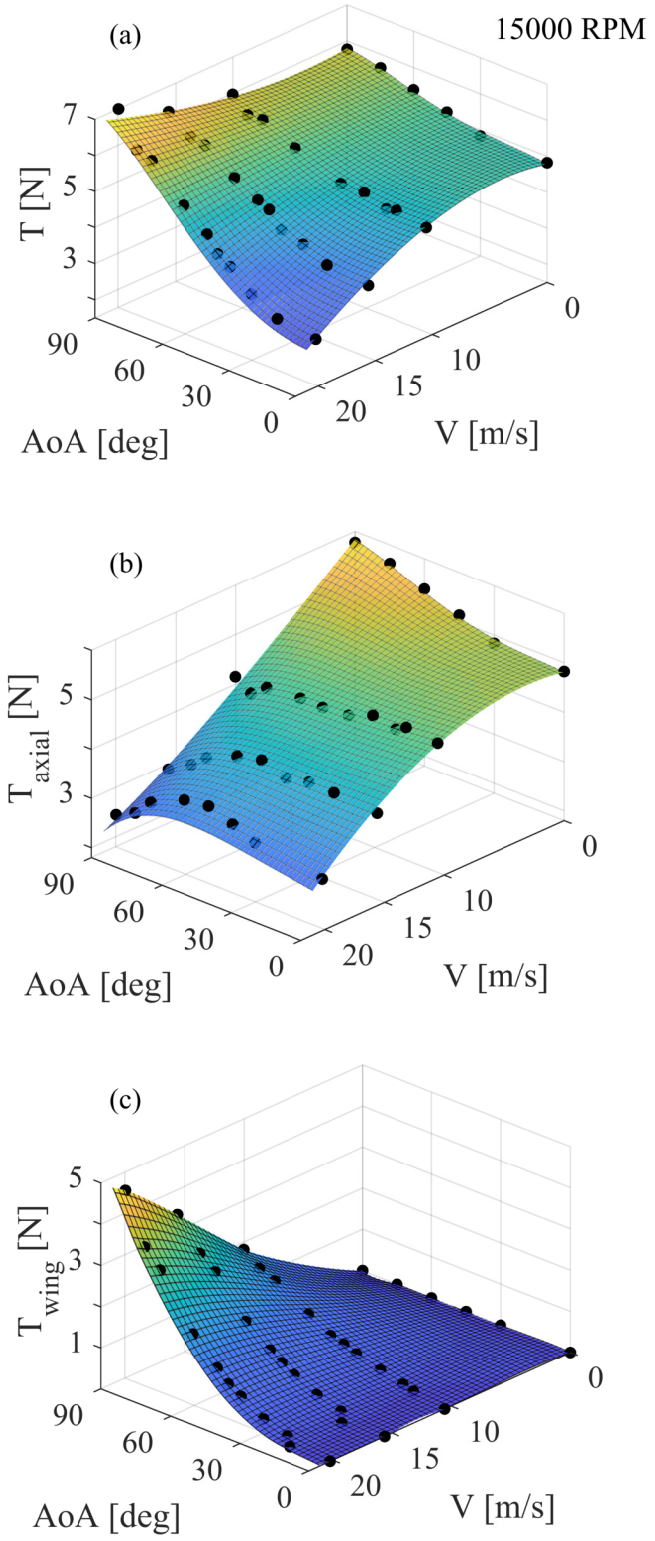


Figure 5.12: Variation of the thrust T, T_{axial}, and T_{wing} with AoA and V: (a) T measured at 15,000 rpm, (b) Calculated T_{axial}, (c) Calculated T_{wing}.

for the cases of fixed velocities $V = 10$ m/s, $V = 15$ m/s, $V = 20$ m/s and RPM values of 9000 and 15000. The increase of thrust with AoA, $(\partial T/\partial \alpha_p > 0)$ follows to a great extent the behavior of T_{wing}, $(\partial T_{wing}/\partial \alpha_p > 0)$ from which it can be inferred the wing component to be mostly the reason to the phenomenon. Also, it can be observed from sub-figures (b), (d), (f) and (a), (c), (e) that as V (and J) is increased, the sensitivity slopes of T_{wing} to AoA are also increased. T_{wing} contribution below $AoA \approx 30^o$ is negligible even at high advance ratios J as the wing factor WF and $\sin \alpha_p$ are low at small angles. At $AoA = 0^o$, T_{wing} vanishes and $T = T_{axial}$, which formula becomes Eq. (2.19) with $e = 1$, where the model reverts to the classic Momentum Theory with no incidence.

T_{wing} relevance starts at $AoA > 30^o$, and a high J value causes the contribution of T_{wing} relative to T_{axial} to become very important in the region around $AoA \approx 60^o$ where T_{wing} eventually surpasses T_{axial} as illustrated in sub-figures (a), (c), (d), (e) and (f). This, according to the theory, should only be observed for w/V lower than around 0.6 (see Fig. 3.5), which is associated with a higher J [see Eq. (2.22b)]. At very high angles and as J is increased, T_{wing} composes an ever larger part of T. Note also that, as J is increased (decreased w/V), the crossing of the two components is possible at a lower angle than 90°, towards 60° [sub-figures (a), (c), (e) and (b), (d), (f)]. This agrees with the theoretical prediction behaviors, as shown in Fig. 3.5.

As T_{axial} is predominant at angles $AoA < 60^o$, T is decreased with increasing speed (and J), and accordingly $(\partial T/\partial V < 0)$. For higher angles, T grows with V, $(\partial T/\partial V > 0)$ [see sub-figures (a), (c), (e) at 9000 rpm and (b), (d), (f) at 15000 rpm]. The inversion from negative to positive sensitivity (see Fig. 5.10), could be explained as the influence of T_{wing} on T, $(\partial T_{wing}/\partial V > 0)$ being predominant over T_{axial}, $(\partial T_{axial}/\partial V < 0)$ in that region. Yaggy and Rogallo, [28, p.18-24], performed several experimental tests for C_T vs J and AoA, ranging from 0° to 85° on three different full scale propellers, at many different blade pitch angles. They found the inversions of slope $(\partial C_T/\partial J)$ occurring at different angles of incidence, for several blade angle configurations, mostly around or above 60°, for blade pitch angles lower than 25^o. As blade pitch angles increased, the inversion of behaviour tended to oc-

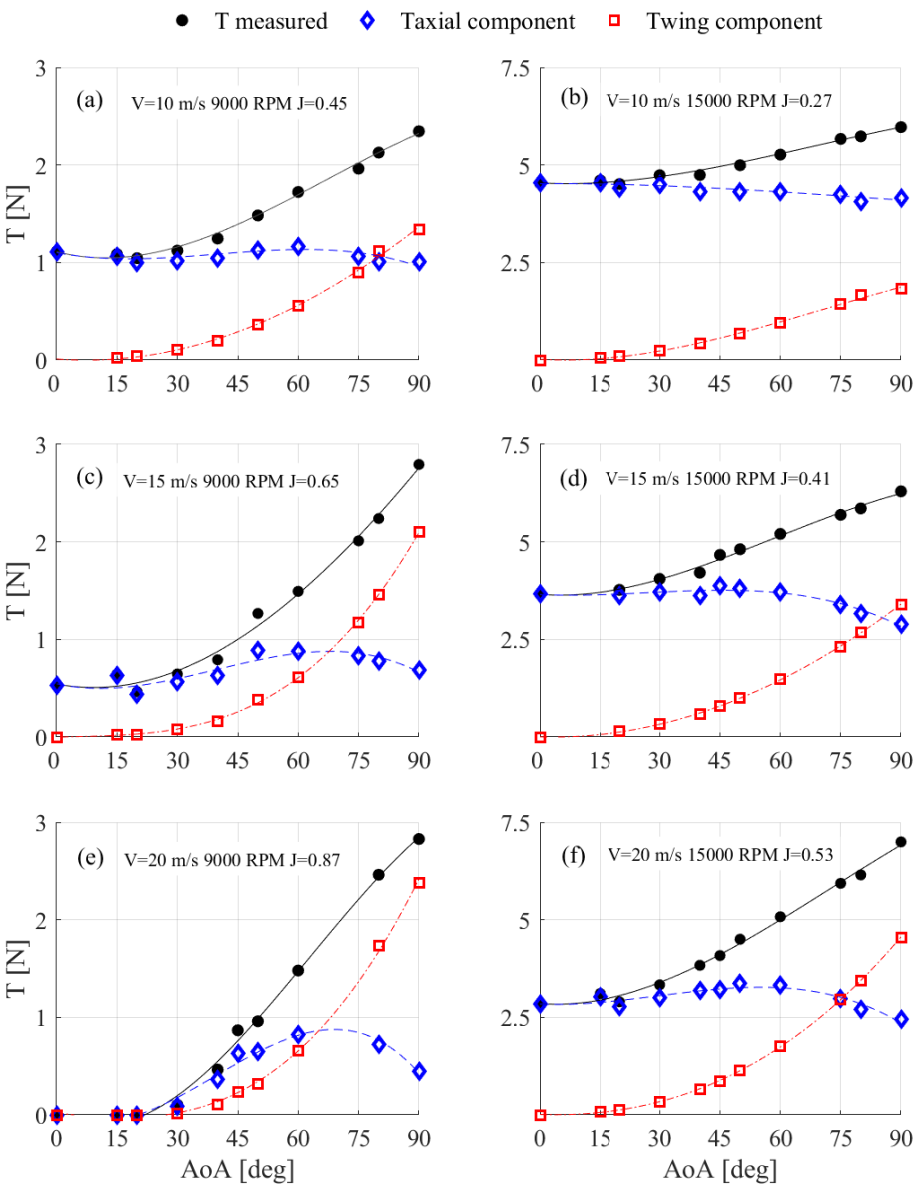


Figure 5.13: T measured, T_{axial} and T_{wing} calculated according to Eqs. (3.4) and (3.8)

cur at lower AoA. It is interesting to note that this inversion is in the region of AoA around where the theory predicts that T_{wing} and its effects begin to overcome T_{axial}, if under a large enough velocity V, meaning high J, (see Fig. 3.5). A possible explanation for the difference in angles, where the inversions occur, would be that different blade configurations would have different slope sensitivities of T to V, $(\partial T/\partial V)$ as different propellers should have different T surfaces vs w/V and

AoA, and so, ($\partial T_{wing}/\partial V > \partial T_{axial}/\partial V$) would happen at different angles, for different propellers. In general, (see [48, p.85, 90, 142],[49, 94, 95]), for blade pitch angles lower than around 25^o, the behaviour inversion occurs at $AoA \approx 60^o$, or higher. However, as seen from the results in [26, 28] it can occur at angles lower than $AoA \approx 60^o$ for higher blade pitch conditions. Still, one should expect necessarily that ($\partial T_{wing}/\partial V > \partial T_{axial}/\partial V$) for the inversion to occur, in the general case of propellers with not exceedingly high blade pitch angles. This will be verified in further tests, in Chapter 6. Nonetheless, for any propeller the relation between T_{wing} and T_{axial} must follow Eq. (3.13), and the component T_{wing} itself can only overcome T_{axial} at angles higher than 60^o and under conditions of $w/V < 0.6$, as illustrated in Fig. 3.5, to fulfil the Momentum Theory assumptions.

T_{axial} is relatively insensitive to AoA at low advance ratios (up to J=0.53 for the propeller tested), and up to angles around 60° [see sub-figures (a), (b), (d), (f)]. A decrease in T_{axial} at higher AoA is noticed in all cases, being more accentuated as J is growing (w/V diminishing), where also T_{wing} growth is more intense. At higher advance ratios, T_{axial} is increased with AoA to peak at around 60° for the propeller studied [see sub-figures (c), (f)].

As T_{wing} tends to be the main component of T at high J and AoA, the thrust formula is indeed reduced to the second term of Eq. (3.12) or $T \rightarrow 2\rho\, V\, w\, S_{disk}$ as predicted by Glauert, [31, p.319], and described in Ref.[42, p.127], for helicopter rotors at high translation speeds (high J), when the propeller behaves as a wing.

5.3.2 Slipstream Analysis

Calculations of the slipstream parameters are presented in the current section. Table 5.1 summarizes the calculation of the relevant angles and velocities, as depicted in Fig. 2.2, from the experimental data. At given AoA, RPM and V conditions and from the thrust T measured, calculations are performed according to Eqs. (2.5b), (2.7), (2.18), (2.21), (2.26), (2.27), (2.28a), or (2.28b), (2.29), (3.4) and (3.8). T_{wing} and T_{axial} are also presented at different flow conditions. For $AoA = 30^o$, the theoretical entrainment factor e is close to unity, as expected from the momentum theory. For

$AoA°$	RPM	V [m/s]	J	w/V	T [N]	T_{axial} [N]	T_{wing} [N]	e	$\varepsilon°$	$\alpha_{slp}°$	$\alpha_{slp\_ult}°$	V_{disk} [m/s]	V_{ult} [m/s]
30	17864	10.4	0.23	0.832	7.001	6.715	0.285	1.0	16.4	13.6	18.8	18.4	26.8
30	14989	15.5	0.41	0.301	4.050	3.723	0.327	1.1	23.2	6.8	11.2	19.7	24.0
30	9024	20.1	0.87	0.006	0.103	0.089	0.014	1.2	29.8	0.2	0.3	20.2	20.3
60	17802	10.5	0.23	0.938	7.668	6.569	1.100	1.2	31.1	28.9	40.0	17.6	26.6
60	15002	15.7	0.41	0.386	5.204	3.721	1.483	1.4	44.4	15.6	25.7	19.5	24.2
60	9074	19.9	0.86	0.081	1.480	0.824	0.655	1.8	56.1	3.9	7.4	20.8	21.7
90	17865	10.3	0.23	1.155	8.289	6.267	2.022	1.3	40.9	49.1	66.6	15.8	26.0
90	14861	15.6	0.41	0.518	6.296	2.896	3.400	2.2	62.6	27.4	46.0	17.6	22.5
90	8979	19.9	0.87	0.160	2.828	0.446	2.382	6.3	80.9	9.1	17.7	20.2	20.9

Table 5.1: T measured, T_{wing}, and T_{axial} calculated from the theory and slip-stream parameters

increasing V values, α_{slp} gets smaller as it is harder for the propeller to turn the flow, especially at low RPM (high J, low w/V), which is in agreement with Ariza, [48, p.123], and if AoA is high, so is e whenever α_{slp} is small. The angle $\alpha_{slp_{ult}}$ is always greater than α_{slp} at the disk as w_{ult} is higher than w (see Fig. 2.2 and Eq. (2.18), $w_{ult} = 2w$). At high AoA, the contribution of T_{wing} becomes more relevant and for w/V values lower than 0.6 and angles higher than 60°, T_{wing} eventually surpasses T_{axial} as in the last two cases where $AoA = 90^o$ and $w/V = 0.518$ and 0.16. Note the entrainment factor e is also high ($e = 2.2$ and $e = 6.3$) accompanied by a high angle ϵ in these instances.

Regarding the relation between V_{ult}, V_{disk}, and V, it can be seen that all the results obtained are in accordance with the theoretical relations determined by equations (3.19) and (3.22).

Figure 5.14(a) depicts the behavior of α_{slp} as a function of J and AoA for the propeller being tested in this work. It is seen that for static tests ($J = 0$), $\alpha_{slp} = AoA$. At no incidence ($AoA = 0^o$), α_{slp} also vanishes as the flow is axial. As J is increased, α_{slp} tends to diminish and for a constant J, α_{slp} grows with increasing AoA. Figure 5.14(b) shows that the angle ϵ is varied with AoA and with J. For static test ($J = 0$), $\epsilon = 0^o$ as w is aligned with T. At $AoA = 90^o$, ϵ tends asymptotically to 90° with the growth of J (which implies growth in V/w).

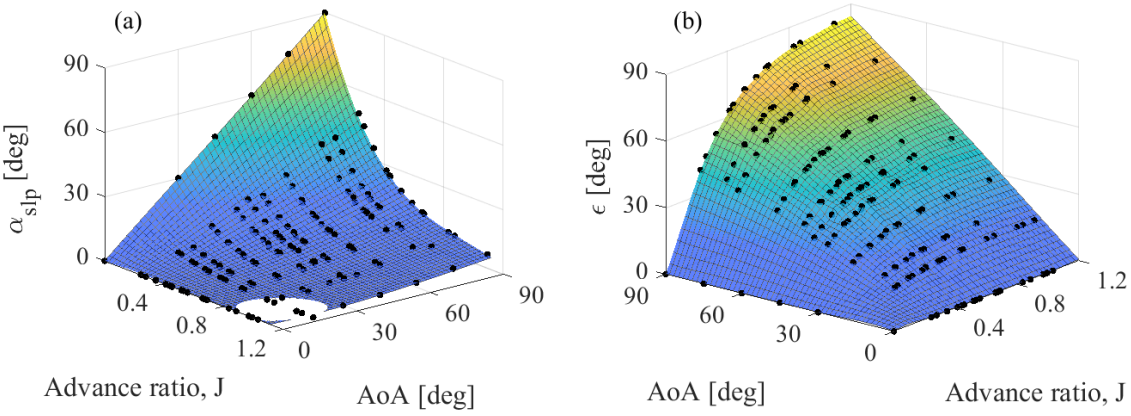


Figure 5.14: Angles variation with J and AoA: a) slip-stream angle at the disk α_{slp}, and b) angle ϵ, between V_{disk} and thrust T

The analysis of ϵ is of interest as it defines the entrainment factor e, the wing factor WF and therefore is directly related to T_{wing}, as indicated in Eqs. (3.8), (3.11) and shown in Fig. 3.1. The maximum ϵ is associated with the highest T_{wing} which occurs at high V, or J (low w/V values) and high angles of incidence [see Fig. 5.12(c)].

5.3.3 Power Measurements Analysis

Electric power measurements are presented in Figs. 5.15(a) and (b) for $RPM = 12000$ and $RPM = 15.000$ tests, respectively. Surfaces fits are also shown, where for Fig. (a), a 3rd degree in AoA, 2rd degree in V polynomial fit with an R^2=0.96 is used. In Fig. (b), a 1st degree in AoA, 3rd degree in V, presenting an R^2=0.93 is fitted to the data. Despite the goodness of fit not being very high for the analysed data, a characteristic behaviour trend of power demand can be observed. Power demand follows thrust produced and the surfaces present similar shape as the thrust surface seen in Fig. 5.12(a). Also, power is higher for the higher RPM operation mode. It is seen that there is an enhancement in power demanded at very high angles AoA and at increasing V, for constant RPM values. Maximum power demanded from the motors, at constant RPM values, occurs at AoA=90° and at higher free-stream speeds V, following thrust behaviour. The explanation for the effect might be given as the contribution of the wing component T_{wing} that enhances thrust.

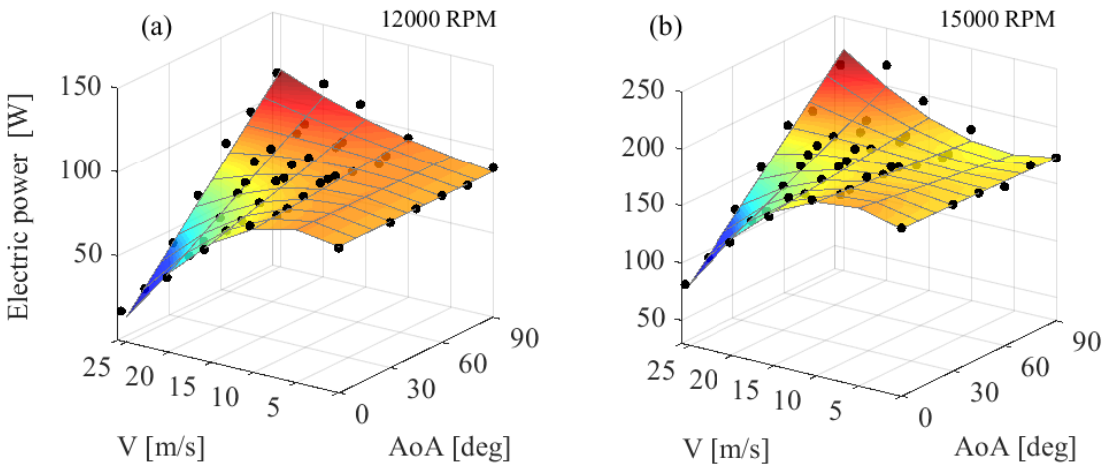


Figure 5.15: Surface fits of electric power measured before ESC, at 12,000 rpm and 15,000 rpm

The thrust enhancement effect, under free-stream speeds and at high angles of attack, enables a reduction in RPM, and power input, while preserving constant thrust. This is especially useful for helicopter rotors in forward flight, as shown in [39, p.215] and in [31, p.311]. An analysis in [38, p.137], explains the effect of reduction in power requirement for a rotor in forward flight, under the scope of blade elements, and the so-called "translation lift" enhancement. In practice, however, there is a limitation to power reduction with translational velocity, as at some point, further V rise will cause the vehicle to experience increase in rotor and fuselage parasite drag, not to mention the reverse flow effect and stall on the retreating blade, at very high speeds.

5.4 A Simplified Formula for Estimating Thrust at Incidence Based on Data at $AoA = 0^o$

5.4.1 Theory Validation

The thrust measurements obtained previously from the wind tunnel experimental tests at any AoA allow us to calculate w/V from Eq. (2.21) and estimate T_{axial} and T_{wing} at any point. However, it is not possible to measure these two components

separately, except in the extreme cases of hover and at high V (high J, low w/V) at $AoA \to 90^o$, where $T \to T_{wing}$ formula according to Eq. (3.12), in accordance to Glauert [31, p.319].

In order to prove the validity of the theory, we use the fact that T_{axial} is relatively insensitive to AoA, unless at high speeds, as seen in Fig. 5.13. The values of thrust measured at $AoA = 0^o$, will be T_{axial} itself, as the wing component vanishes at no incidence. Therefore, using $T|_{AoA=0}$, and assuming it to be the axial component for all angles, it is possible to rewrite Eq. (3.4) that defines T_{axial} at any angle, as:

$$T_{axial} = (2\rho\, S_{disk}\, V \cos\alpha_p)\, w + (2\rho\, S_{disk})\, w^2 \approx constant \approx T|_{AoA=0} \tag{5.3}$$

which allows for the estimation of an approximate projection of the induced speed at any AoA, based only on the thrust measured at $AoA = 0^o$. As only non-negative w values are considered, solving Eq. (5.3), leads to:

$$\left(\frac{w}{V}\right) \approx \left(\frac{w}{V}\right)_{approx} = \frac{1}{2}\sqrt{\cos^2\alpha_p + \frac{2\,T|_{AoA=0}}{\rho\, S_{disk}\, V^2}} - \frac{\cos\alpha_p}{2} \tag{5.4}$$

Equation (5.4), can also be written as:

$$\left(\frac{w}{V}\right) \approx \left(\frac{w}{V}\right)_{approx} = \frac{1}{2}\sqrt{\cos^2\alpha_p + \frac{8\,C_{T|_{AoA=0}}}{\pi\, J^2}} - \frac{\cos\alpha_p}{2} \tag{5.5}$$

Fig. 5.16 illustrates w/V calculated from real measured T data at all angles, against the approximation model $(w/V)_{approx}$ in solid lines, based solely on $T|_{AoA=0}$. It is seen that there is a good match of the simplified model with the measurement data down to $w/V = 0.2$ or J up to 0.53. For $J > 0.53$ there is a detachment past $AoA = 30^o$ that grows with J. At $J = 0.87$ and $AoA = 0^o$, the propeller is windmilling and $T|_{AoA=0^o} = 0$ as in Fig. 5.13 (e) so, the model is not suitable as it predicts $w = 0$ for all angles.

Fig. 5.17 presents the results obtained from actual thrust measurements against predicted T through the simplified model according to Eq. (3.16), using the values

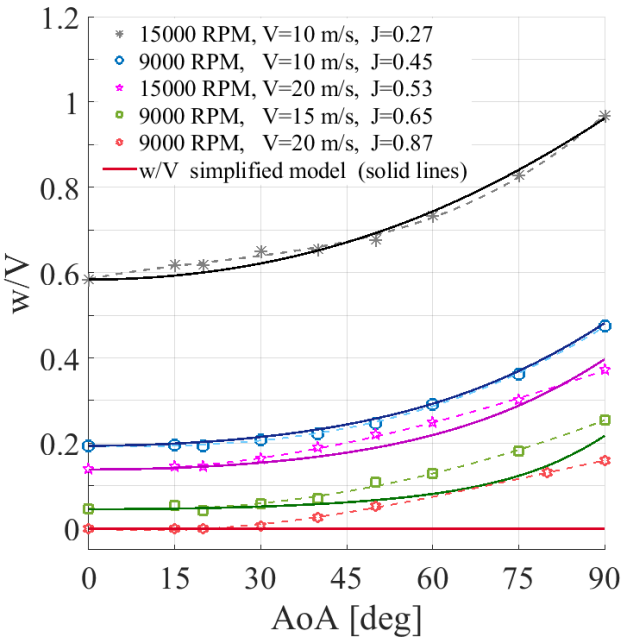


Figure 5.16: w/V obtained from T data at AoA compared to w/V estimated from $T|_{AoA=0^o}$ and extrapolated to other AoA values (solid lines)

of T at $AoA = 0^o$. Also w/V projected to be used in Eq. (3.16), is obtained using T at $AoA = 0^o$, according to Eqs. (5.4), or (5.5), as illustrated in Fig. 5.16.

As the simplified model relies on a projection of thrust at $AoA = 0^o$ and as T_{axial} is relatively constant up to AoA around 60^o, at low advance ratios (see Fig. 5.13 for $J < 0.53$), a good agreement is observed between the model and the actual thrust [see Fig. 5.17 (a), (b), (d)]. In those tested cases, w/V projected from the simplified model is very close to w/V obtained from experimental data (see Fig. 5.16). At around $AoA > 70^o$, the model overestimates the practical test data. This should be expected as the model assumes a T_{axial} constant for all incidence angles but it can be observed from Fig. 5.13 (a), (b) and (d) a decrease in that component at a higher AoA. A small detachment of w/V projected by the model, from measured w/V is seen at $J = 0.53$ in Fig. 5.16, which will cause the start of the detachment of T estimated by the model from the real T measured, as in Fig. 5.17 (f). As J grows, the detachment of T projected by the model increases as in Fig. 5.17(c) until the model eventually loses validity as in Fig. 5.17 (e). This happens when $T|_{AoA=0^o}$ measured at no incidence is vanished (windmill/brake state), alongside w/V, that is

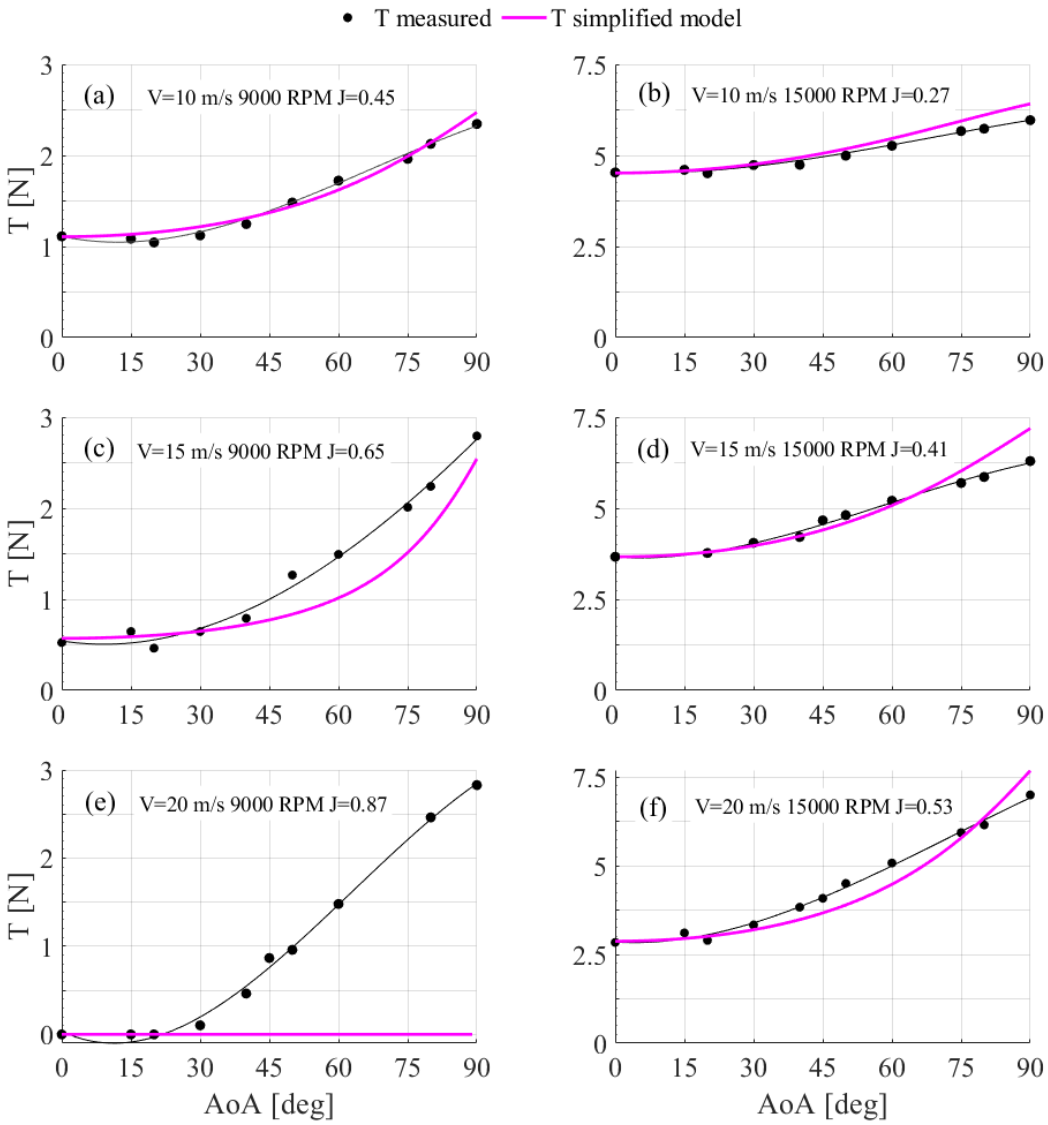


Figure 5.17: T measured vs simplified model projected from $T|_{AoA=0^o} = 0$ on Eqs. (2.21) and (3.16)

used by the model to project T at incidence. The growing detachments of the model w/V, from w/V calculated based on the real data, occur in consonance with T_{axial} no longer being relatively constant and similar to $T|_{AoA=0^o}$ at angles up to around 60^o, for increasing J values. In these cases, the T_{axial} growth with AoA is not captured by the simplified model that underestimates T up to those angles.

5.4.2 The Simplified Thrust Formula

The simplified formula for thrust prediction at incidence, used to produce Fig. 5.17 in the previous section 5.4.2 is presented below:

$$T = T|_{AoA=0^o} \left[1 + \frac{(\sin\alpha_p)^2}{\left[\cos\alpha_p + \left(\frac{w}{V}\right)\right] \left[\sqrt{1 + 2\left(\frac{w}{V}\right)\cos\alpha_p + \left(\frac{w}{V}\right)^2} + \cos\alpha_p + \left(\frac{w}{V}\right)\right]} \right] \tag{5.6}$$

where the value of w/V used is $(w/V)_{approx}$, obtained from solving Eqs. (5.4), or (5.5), while utilizing the same thrust data acquired at $AoA = 0^o$, i.e. ($T_{axial} = T|_{AoA=0^o}$).

The formula assumes that T_{axial} is relatively constant with AoA up to around 60^o, for intermediate J values. Here, intermediate J are considered to be values where the propeller operates relatively far from the region where T declines substantially to approach the windmill state.

5.5 Concluding Remarks

This initial round of tests produced results that according to the developed theory allowed for the calculations of the axial and wing equivalent thrust components, T_{axial} and T_{wing}. Despite the impossibility to measure either one of these components individually, the thrust behaviour showed promissory results, as the inversion of sensitivity to V ocurred at high angles, as predicted by the theory. Also, calculated w, T_{axial}, and T_{wing} demonstrated a peculiar behaviour, that allowed under certain circumstances, (low J), for a reasonable prediction of thrust at angles of incidence, based on the performance at $AoA = 0^o$. This would only be possible if the existence of the two thrust components were indeed correct. The assumptions of low J permitted the creation of a simplified model for the prediction of T at angles of incidence from data at $AoA = 0^o$, which is used to validate the model.

For the propeller studied, it was seen that at a given RPM, T_{axial} is decreased

with increasing airspeed ($\partial T_{axial}/\partial V < 0$), for all angles of incidence. T_{axial} is shown to be not so much sensitive to AoA at low airspeeds, especially at $AoA < 60^o$, while at high airspeeds it grows with the angle of incidence up to $AoA \approx 60^o$. At higher angles, T_{axial} then decreases with increasing AoA. This decrease is more intense at higher speeds. T_{wing} was found to rise with AoA, and with V at high angles. Also, the sensitivity slope of T_{wing} to AoA increases with V.

Thrust is found to be decreased with V, ($\partial T/\partial V < 0$) at low angles, as T_{axial} is dominant, whereas at around $AoA \approx 60^o$ or higher, and at high airspeeds, T changes its behavior to increase with increasing V, i.e. ($\partial T/\partial V > 0$). This behavior is interpreted as being the consequence of T_{wing} impact overcoming the contribution of T_{axial} effects.

As T increases with V at high angles, a reduction in RPM and power requirements is possible, while maintaining the same thrust output. This enhancement in thrust and the potential reduction of power is attributed, here, to the T_{wing} component.

Different propellers and blade configurations should present different surfaces of T vs w/V and AoA. Therefore, different slope sensitivities ($\partial T_{wing}/\partial V$), ($\partial T_{axial}/\partial V$) and $\partial/\partial V(\partial T_{wing}/\partial \alpha_p)$ should be expected, for a given RPM. It is believed that this could explain the inversion of $\partial T/\partial V$ at different angles of incidence for different propellers seen in other studies. However, the ratio T_{wing}/T_{axial} must follow the theory and the inversion of thrust behavior should happen around the region where T_{wing} becomes relevant and ($\partial T_{wing}/\partial V$) overcomes ($\partial T_{axial}/\partial V$), which happens at high angles and at high speeds ($w/V < 0.6$). This verification is done in the second round of experimental tests, which results are presented in the next Chapter 6.

Chapter 6

Further Experimental and Theoretical Investigation

6.1 Introduction

In light of the results obtained in the first round of tests, a second round was prepared to investigate the validity of the theory for other propellers. Specifically, the behaviour of T_{wing}/T_{axial} against w/V for all angles of incidence, that according to the theory must follow Eq. (3.15), as illustrated in Fig. 3.5, for all propellers. However, the behaviour of T_{wing}/T_{axial} against J for different propellers should have distinct outputs, according to the specific curve of w/V x J, that is expected to be dependent on blade geometry and characteristics of each propeller. This is done in following sections, where also inspections of w/V x J fits behaviour for the propellers tested are presented to permit the referred analysis.

Following the results obtained in Chapter 5, the new tests aim to investigate what is the role of the sensitivities $(\partial T_{axial}/\partial V)$, and $(\partial T_{wing}/\partial V)$ on the inversion of thrust behavior, at high angles, shown in Chapter 5. As $T = T_{axial} + T_{wing}$, and so, $\partial T/\partial V = \partial T_{axial}/\partial V + \partial T_{wing}/\partial V$, it is expected that the transition angle occurs when $(\partial T_{wing}/\partial V)$ begins to overtake $(\partial T_{axial}/\partial V)$, as the axial term in general is reduced with V, and the wing term grows with V for increasingly higher angles. The adding result of opposing component sensitivities should dictate the total thrust behaviour. The analysis is presented, this time, in terms of C_{Taxial}, C_{Twing}, and the advance ratio J, instead of T_{axial}, T_{wing}, and V.

6.2 2nd Campaign Test Rig Configuration and Setup

In the second round of tests, the turntable underneath the test section of the wind tunnel was available, and could be utilized to set the angles of incidence with no limitations. Another aluminum rig, was then manufactured to hold a 10 mm square steel rod setup with a 3D printed motor-propeller assembly. Figure 6.1 shows the rig manufactured for the second round of tests.

Four fixed-pitch propellers were tested, with the following characteristics:

Name definition	no. of blades	D, [in]	pitch, [in/rev]	$\beta_{0.75}$, [deg]
APC 18x5.5"	2	18	5.5	7.4
APC 15x4"	2	15	4	6.4
AEOrc 10x10"	2	10	10	23
MAS 3B-12x6"	3	12	6	12

Table 6.1: Description of the four propellers tested in the 2nd round of tests

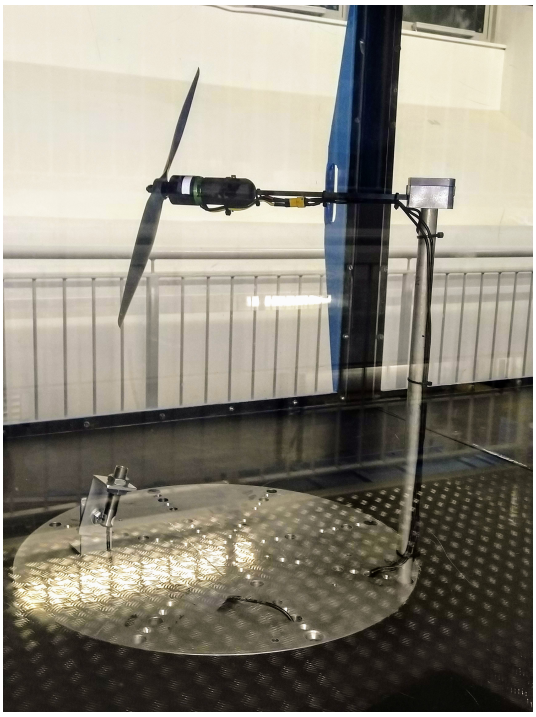


Figure 6.1: New Rig utilized in the 2nd round of tests

The APC propellers were manufactured by "Advanced Precision Composites Propellers" [96], the AEOrc propeller was supplied by a hobby shop [97], and the 3 bladed propeller, produced by "Master Airscrew Propellers" [98].

Two motors were utilized in the tests. A Cobra 2826/10 KV920, by "Cobra Motors" [99], and for the larger propeller, an Axi 2835/10 KV690 that provides higher torque, by "AXI motors" [100]. The same ESC and optical LED sensor for RPM measurements, utilized in the initial tests, were reused here. Figures 6.2a, 6.2b, and 6.2c show the propellers, and the two motors, utilized in the experiments.

(a) The 2-bladed propellers utilized in the tests: 18x5.5 ", 15x4 ", 10x10 "

(b) The 3-bladed MAS 3B-12x6" propeller

(c) The 2 motors utilized in the tests

Figure 6.2: The propellers and motors utilized in the tests

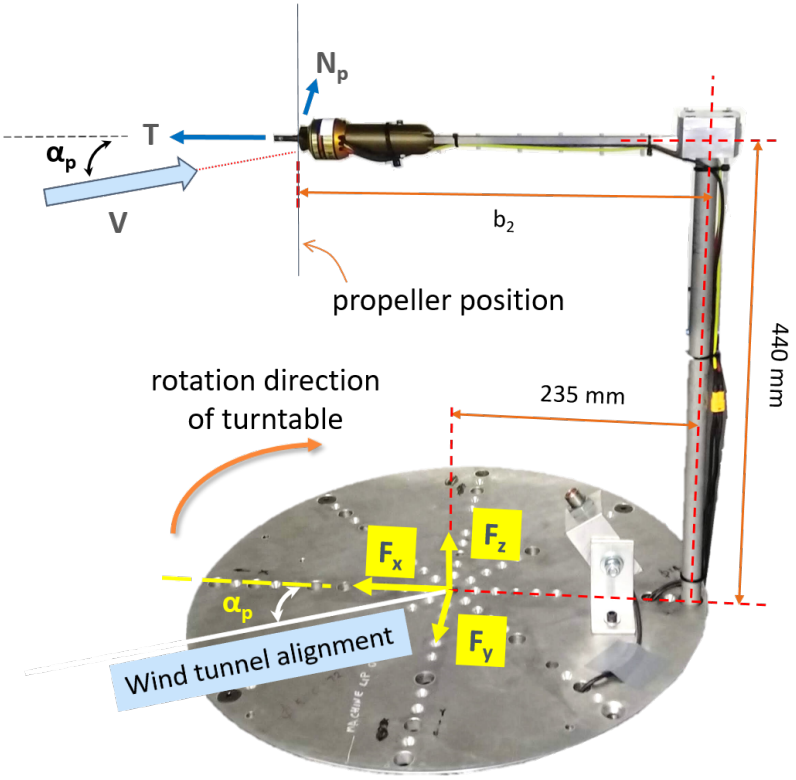


Figure 6.3: Scheme of propeller forces for the second wind tunnel tests rig.

Figure 6.3 depicts the scheme of forces for the new rig in the second round of tests, where the length b_2 is dependent on the motor utilized, being 320 mm for the Cobra, and 360 mm for the AXI. As the turntable is aligned with the force-balance sensor and with the propeller horizontal holding stick, thrust is acquired directly from the readings in the x-axis direction, that are netted from the the rig resistance force, which is available through the function fits that were obtained previously from wind tunnel tests without the propeller. Propeller normal force due to incidence is perceived in the y-axis direction readings, according to the following equations:

$$T = F_x \tag{6.1a}$$

$$N_p = -F_y \tag{6.1b}$$

where F_x and F_y are the read forces in x and y directions, net of rig resistance.

Propeller thrust, and normal forces readings were acquired at wind tunnel speeds ranging from zero to 20 m/s, and motor speed rotations from 3,000 to 9,000 rpm, with different ranges for each propeller. Again, the study here is focused only on thrust analysis, and so N_p analysis is not discussed. Angles of attack ranged from 0° to 90°. For the APC 18x5.5" propeller, rotation speed tests were aimed at around 3,000, 4,000, 5,000, and 6,000 rpm, and wind speeds V aimed at 0, 10, 15, 20 m/s. A total of 60 test conditions were evaluated. For the APC 15x4" propeller, rotation speed tests were aimed at around 3,000, 5,000, 7,000, and 8,000 rpm, and wind speeds V aimed at 0, 10, 13, 17 m/s. A total of 86 test conditions were evaluated. For the 10x10" propeller, 60 points were gathered, being RPM aimed at 4,000, 6,000, 8,000, and 9,000 rpm, and V at 0, 10, 15, 20 m/s. For the MAS 3B-12x6" propeller, 80 conditions were evaluated, at RPM aimed at 4,000, 6,000, and 8,000 rpm and V at 0, 10, 15, 20 m/s. Some were outliers and ignored in the results analysis.

6.3 Evaluation of Results Errors

The evaluation of errors was performed according to the calculations presented previously in Section 4.3. The resulting graphs are displayed in appendix A. Figures A.1 and A.2 present the errors in the experiments for propeller APC 18x5.5". Figures A.3 and A.4 show the tests errors for propeller APC 15x4", while Figs. A.5 and A.6, the errors in the experiments for propeller AEOrc 10x10", and finally, Figs. A.7 and A.8 depict errors for the 3 bladed propeller MAS 3B-12x6" experiments.

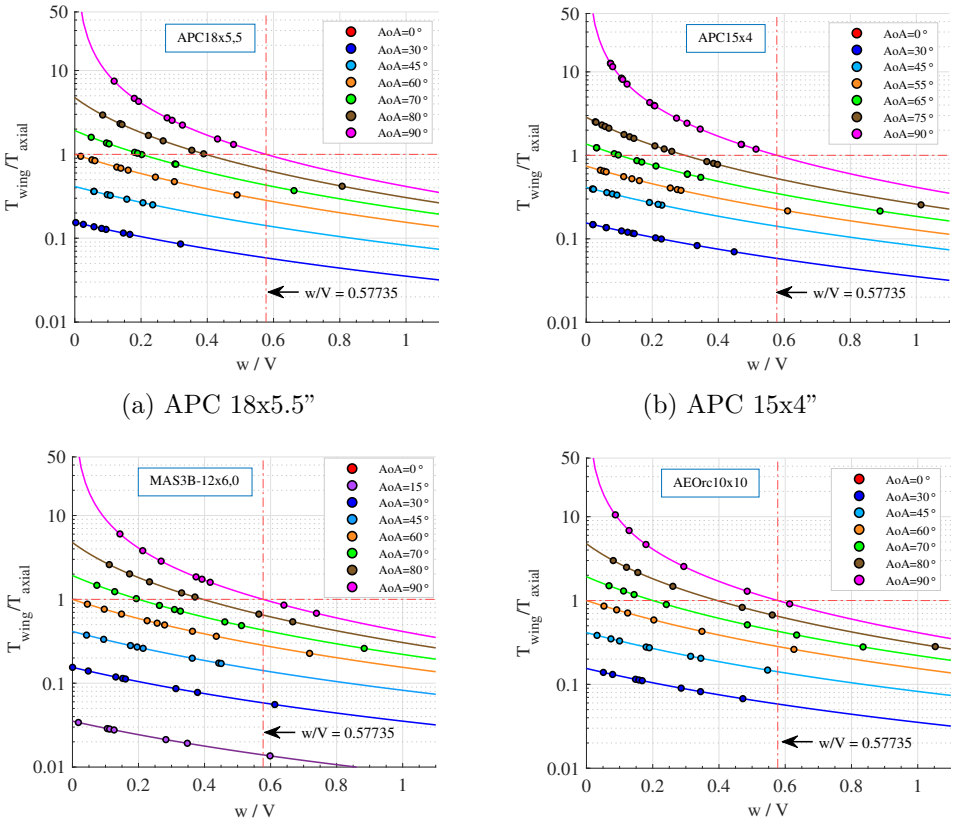
The percentage error analysis disregards the high errors outcomes measured when the variable presents very low average value, close to zero. The cumulative distribution functions, next to each error plot, allows for an evaluation of the quality of the data. For the four propellers, the percentage error in V was lower than 5% in 90% of the cases. A few points, mostly at lower speed tests presented very high error, reaching almost 20%. The same analysis is valid for the percentage error in J, which shows less than 5%, in the vast majority of cases. For RPM, most errors lied below 2%. For thrust T, results show errors almost always lower than 5%. For w, most of the experiments had errors lower than 3%, while for C_T, lower than 2.5%.

6.4 Experimental Results and Discussion

6.4.1 T_{wing}/T_{axial} vs. w/V

In this section we aim to investigate the behaviour of the ratio T_{wing}/T_{axial}, for all the propellers at different angles of attack, and wind velocities. The thrust measurements from the experiments allows for the calculations of w, according to Eq. (2.21). Then, using the known value of w/V on Eq. (3.15), the ratio T_{wing}/T_{axial} can be calculated. The results of the tests in this section are used to show that a direct consequence of the theory is that T_{wing}/T_{axial} relation with w/V is propeller independent.

Figure 6.4 shows all the points obtained in the tests for all propellers, in logarithmic scale for ease of visualization. It can be seen that regardless of the propeller utilized, the values of w/V must follow the theory and lie on the lines of the surface,



(a) APC 18x5.5”

(b) APC 15x4”

(c) MAS 3B-12x6”

(d) AEOrc 10x10”

Figure 6.4: T_{wing}/T_{axial} as a function of w/V

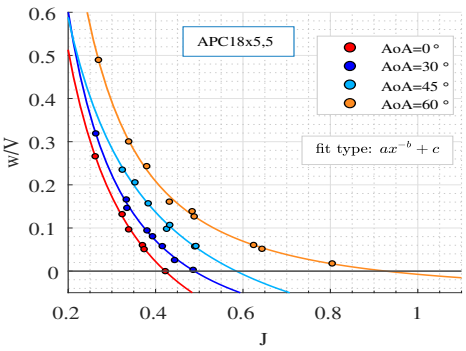
predicted by Eq. (3.15), and plotted in Fig. 3.5. The ratio T_{wing}/T_{axial} may reach unity only at certain combinations of w/V and AoA values, and only at angles higher than 60^o. For instance, at $AoA = 90^o$, it happens at $w/V = 0.57735$, meaning that increasing the wind speed V past that point, will cause T_{wing} and its effects to be dominant on the behaviour of the propeller. At lower angles, a requirement of higher speeds is necessary for T_{wing} to overcome T_{axial}. At $AoA = 80^o$, $w/V < 0.4$ will suffice for the overtaking to happen. At $AoA = 70^o$, it only can happen at $w/V \approx 0.2$ or higher , and below $AoA = 60^o$ it never occurs, for any propeller.

It is also worth stressing that along the ordinate axis, where $w/V = 0$, it is to be expected that the propeller will be in windmill state, if AoA is not sufficiently high. In those conditions, no thrust is produced and both thrust components must go to zero, as $T_{axial} \to 0$, causes $T_{wing} = T_{axial}(T_{wing}/T_{axial}) \to 0$. Now, recalling the results in chapter 5, at some angle higher than 60^o, it is also to be expected that propellers will start behaving as wings, meaning that even at low w/V values, there will be production of thrust, and the values of T_{wing}, and T_{axial} do not vanish near the ordinate axis. At very high angles, $AoA \to 90^o$, and at high V, or high J, $T_{wing}/T_{axial} \to \infty$, as $w/V \to 0$.

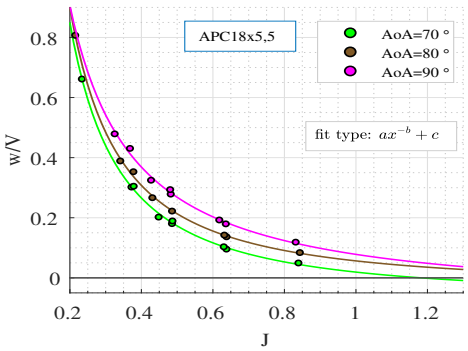
6.4.2 w/V vs. J fits

As T_{wing}/T_{axial} behaviour against w/V is the same for all propellers, in accordance to the theory and shown above, then the particular characteristics of each propeller must be noticeable in terms of other parameters. Propellers of different pitch, and distinct blade geometry can be evaluated in terms of the advance ratio, J, influence. Next, an analysis of the ratio w/V against J is presented in Fig. 6.5, for the 4 propellers tested. The inverse relation of J with w/V is noticeable, as V influences the two parameters in an opposite way. Curve fits are performed on the data obtained in the tests. The fits will be later utilized for further calculations of T_{wing}/T_{axial} against J.

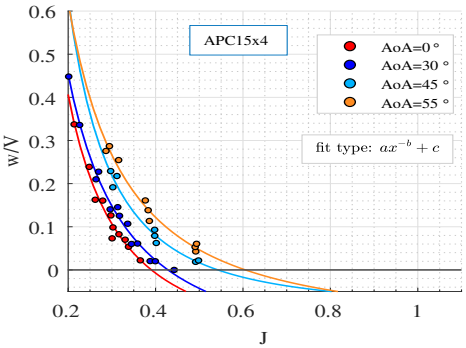
Now, it can be seen that every propeller behaves differently, for different J values. When the curves cross the abscissa at $w/V = 0$, thrust is not produced and the propeller operates at windmill state. For the higher pitch propellers, the crossing



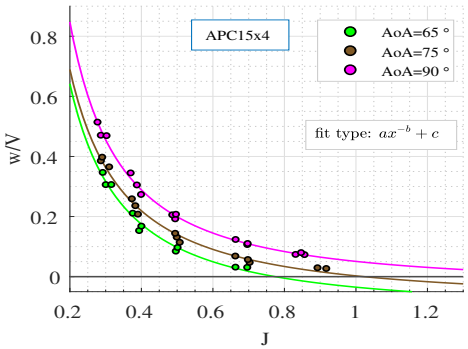
(a) APC 18x5.5", $AoA \leq 60^o$



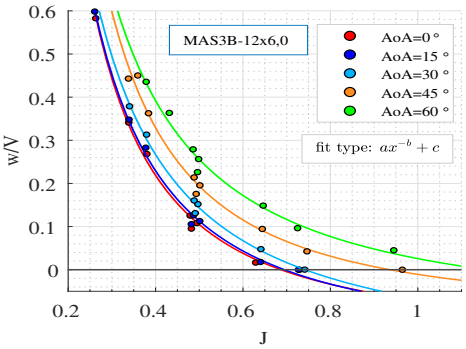
(b) APC 18x5.5", $AoA > 60^o$



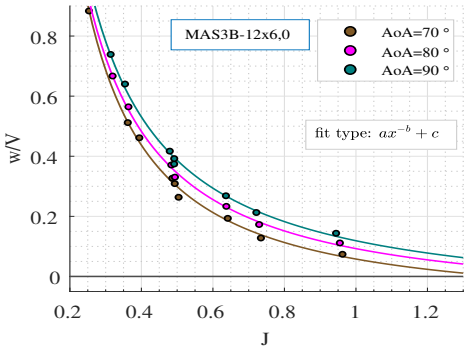
(c) APC 15x4", $AoA \leq 60^o$



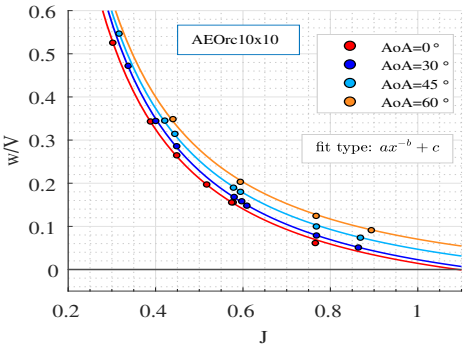
(d) APC 15x4", $AoA > 60^o$



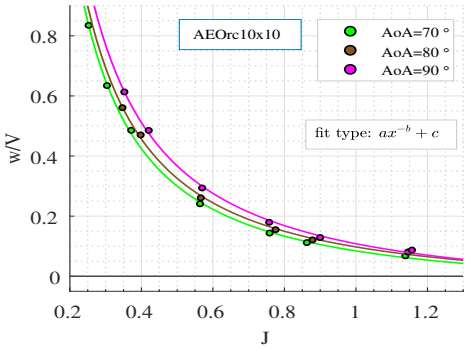
(e) MAS 3B-12x6", $AoA \leq 60^o$



(f) MAS 3B-12x6", $AoA > 60^o$



(g) AEOrc 10x10", $AoA \leq 60^o$



(h) AEOrc 10x10", $AoA > 60^o$

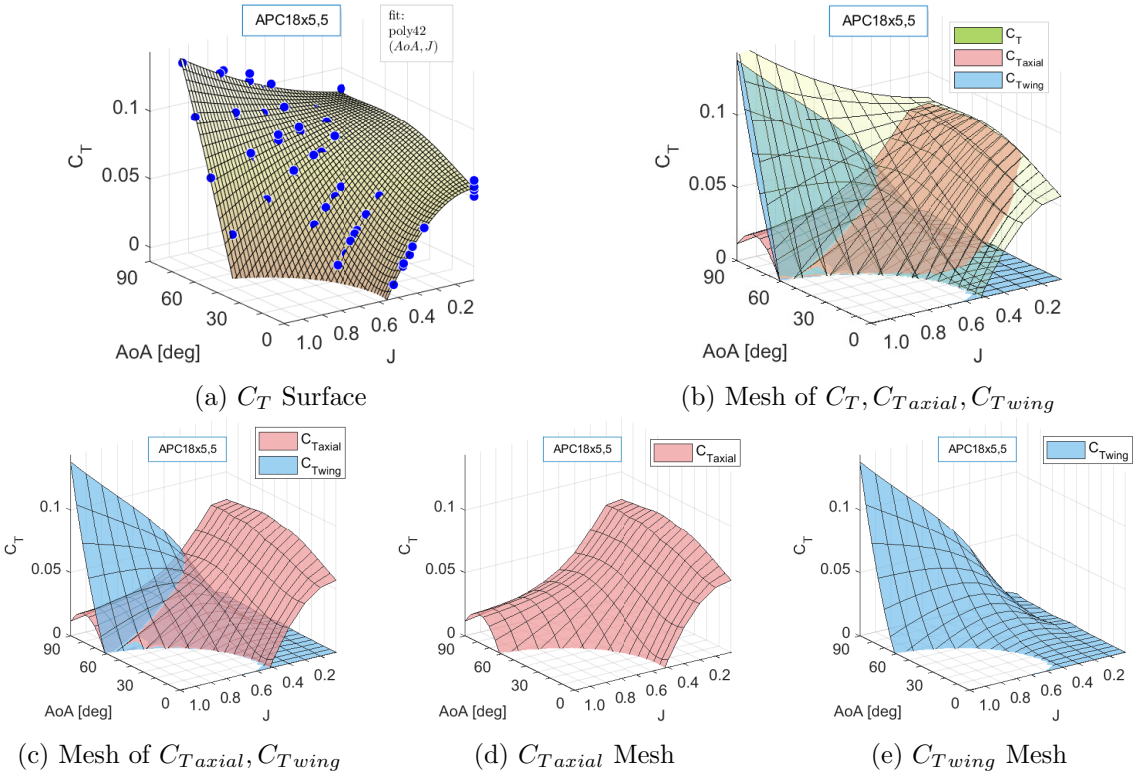
Figure 6.5: Fits of w/V as function of J, for the propellers tested

of $w/V = 0$ will happen at higher J values, meaning that higher pitch propellers can still deliver thrust at higher wind speeds, where low pitch propellers are already at windmill state. For all propellers, the increase in AoA will cause the crossing to happen at higher J values. Eventually, at very high angles, w/V asymptotically decreases with J increases, and never crosses the zero line. In these cases, the influence of T_{wing} is much higher than T_{axial}, as seen from Fig. 6.4, and thrust is always being produced, and even increases with speed. (It is important to stress that the analysis here is limited to speeds lower than where retreating blade stall effects occur). This behavioural transition will happen at different angles for the distinct propellers tested, and according to the curve fits, always at $AoA > 60^o$. For instance, for the low pitch propeller APC 15x4", in Fig. 6.5d, it is seen that there is still a crossing to windmill state, at $AoA = 75^o$, which occurs at around $J \approx 1$, so the expected behavioural transition should happen at an even higher angle. In Fig. 6.5h, the higher pitch propeller AEOrc 10x10" already does not display a crossing of the zero line, apparently from $AoA \geq 70^o$, and $J > 1.2$. At $AoA = 90^o$, the crossing never happens, for all propellers.

6.4.3 Surfaces of Thrust Coefficients

In order to further analyse the characteristics of thrust behaviour, with respect to J, and AoA, surfaces of C_T, C_{Taxial}, and $C_{T_{wing}}\, vs\, J$, and AoA were produced, for all the propellers tested.

The results of $C_T\ vs\ J$, and AoA, measured from all tests, for propeller APC 18x5.5", are presented in Fig. 6.6a. A surface, based on a polynomial of 4th degree in AoA, and 2nd degree in J, is fitted to the points, having an adjusted $R^2 = 0.9911$ and $rmse = 0.0030$. From the polynomial fit, a meshed plot surface for C_T is produced for several pre-defined values of AoA, and J, by using the function "mesh" in MATLAB. Then, the ratio T_{wing}/T_{axial}, as function of AoA, and J, is calculated for all points defined by the mesh, through Eq. (3.15), where an interpolated fit function $w/V = f(AoA, J)$ is used as input. From the C_T surface, and the T_{wing}/T_{axial} ratio, it is possible to obtain, first, the surface for C_{Taxial}, according to $C_{Taxial} = C_T/(1+T_{wing}/T_{axial})$, and then, the C_{Twing} surface, as



(a) C_T Surface

(b) Mesh of $C_T, C_{Taxial}, C_{Twing}$

(c) Mesh of C_{Taxial}, C_{Twing}

(d) C_{Taxial} Mesh

(e) C_{Twing} Mesh

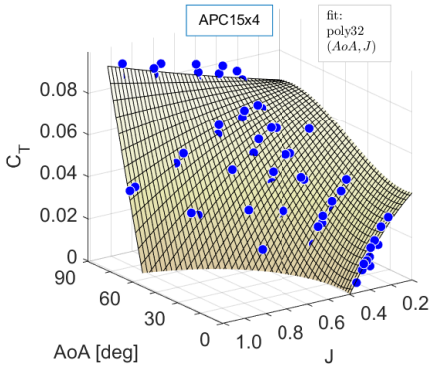
Figure 6.6: Surfaces of Thrust Coefficients for Propeller APC 18x5.5”

$C_{Twing} = C_T - C_{Taxial}$. The C_T, C_{Taxial}, C_{Twing} interpolated surfaces are necessary for plotting any arbitrary J and AoA conditions chosen, which will be required for the analysis in the following sections, regarding the inversion of thrust behaviour, at high angles, as J grows. Figure 6.6b depicts the meshed C_T, and the underlying C_{Taxial}, C_{Twing} surfaces.

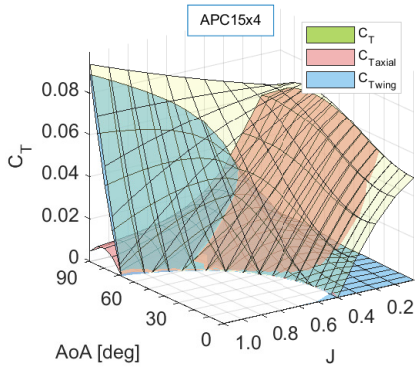
The same analysis is applied to the remaining propellers. Figure 6.7a illustrates the results for propeller APC 15x4”. The surface fit to the points is based on a polynomial of 3th degree in AoA, and 2nd degree in J, having an adjusted $R^2 = 0.9751$, and $rmse = 0.0037$. The results of the tests, for propeller MAS 3B-12x6”, are presented in Fig. 6.8a, where a polynomial of 4th degree, in AoA, and 4th degree in J is used to the surface fit, with an adjusted $R^2 = 0.9781$ and $rmse = 0.0070$. Finally, the results for propeller AEOrc 10x10”, are presented in Fig. 6.9a. The surface fit, there, is based on a polynomial of 2nd degree, in AoA, and 2nd degree in J, and has an adjusted $R^2 = 0.9834$, and $rmse = 0.0042$. The choice of the polynomial degree was made based on the best fit possible. It is worth noting that at low J, not much

experimental data was available, but only a few points from static tests, for what the fits might not be ideal at low J. Also, as in the 1st round of tests in Chapter 5, an increase in the measured thrust at high angles and low speeds (or low J) is noticeable when comparing to results at low angles, which is again believed to be due to wind tunnel wall effects. Figures 6.7b, 6.8b, 6.9b, present the meshed C_T surfaces, and the underlying C_{Taxial}, C_{Twing} surfaces for the three other propellers, APC 15x4", MAS 3B-12x6", and AEOrc 10x10".

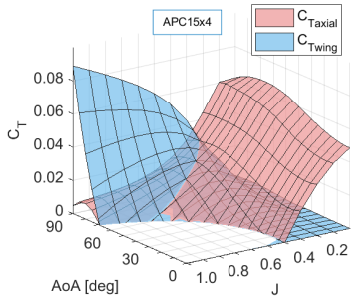
The two components of C_T are shown together, for all propellers in Figs. 6.6c, 6.7c, 6.8c, 6.9c, where it can be seen that C_{Taxial} is dominant at lower angles and low J, whereas C_{Twing} rules at angles starting from $AoA > 60^o$, and increasing J. The border region, where the latter overtakes the previous, depends on J, and AoA, and shown to be different for each propeller. For higher J values, the overtaking of C_{Twing} over C_{Taxial} can happen at diminishing angles AoA, from 90^o towards 60^o, but always



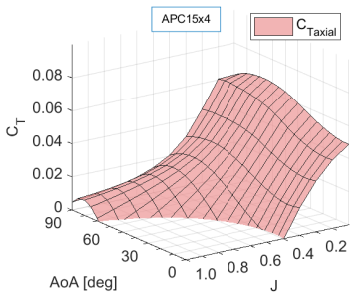
(a) C_T Surface



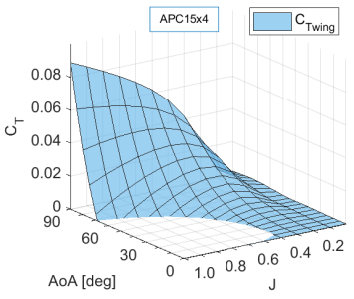
(b) Mesh of $C_T, C_{Taxial}, C_{Twing}$



(c) Mesh of C_{Taxial}, C_{Twing}

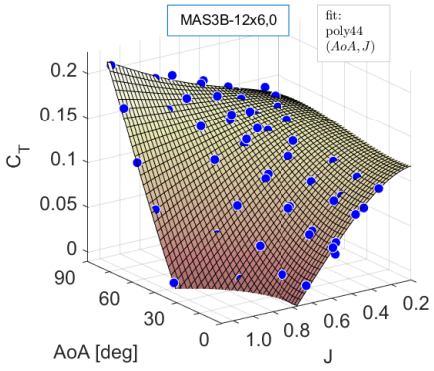


(d) C_{Taxial} Mesh

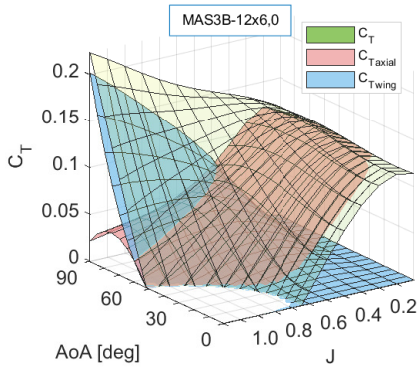


(e) C_{Twing} Mesh

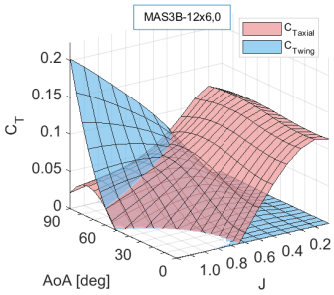
Figure 6.7: Surfaces of Thrust Coefficients for Propeller APC 15x4"



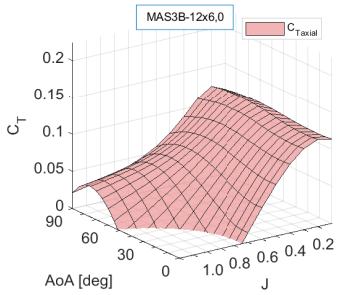
(a) C_T Surface



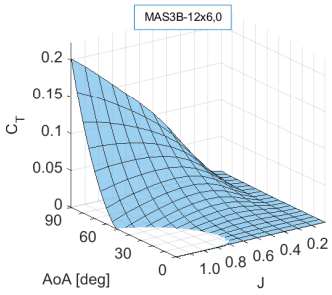
(b) Mesh of $C_T, C_{Taxial}, C_{Twing}$



(c) Mesh of C_{Taxial}, C_{Twing}

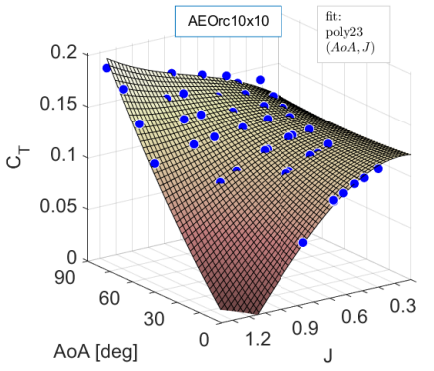


(d) C_{Taxial} Mesh

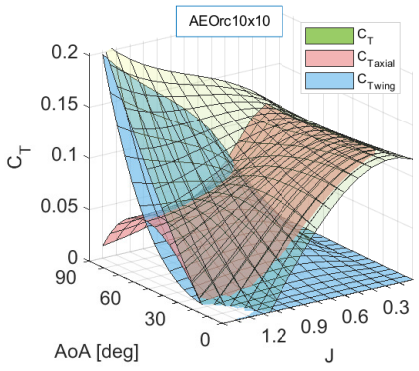


(e) C_{Twing} Mesh

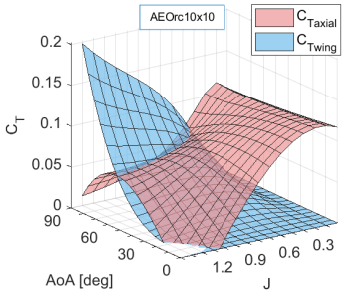
Figure 6.8: Surfaces of Thrust Coefficients for Propeller MAS 3B-12x6"



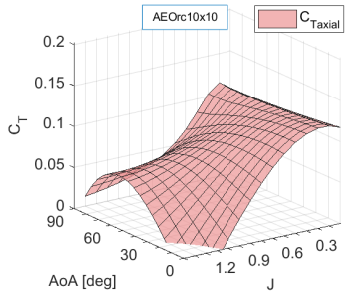
(a) C_T Surface



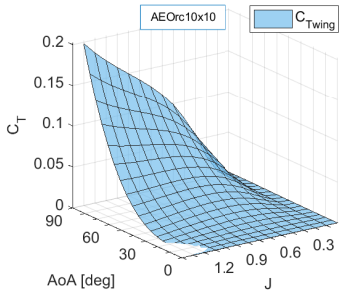
(b) Mesh of $C_T, C_{Taxial}, C_{Twing}$



(c) Mesh of C_{Taxial}, C_{Twing}



(d) C_{Taxial} Mesh



(e) C_{Twing} Mesh

Figure 6.9: Surfaces of Thrust Coefficients for Propeller AEOrc 10x10"

at $AoA > 60^o$ as imposed by the theory, and seen in Figs. 6.4, and 3.5.

Figures 6.6d, 6.7d, 6.8d, and 6.9d show the C_{Taxial} component of each propeller, where it is seen that C_{Taxial} always diminishes with J. This reduction appears to be in an asymptotic way, at high angles, and high J. Also, as AoA, and J grow, C_{Taxial} appears to grow, to then recede with AoA.

The C_{Twing} component of each propeller appears in Figs. 6.6e, 6.7e, 6.8e, 6.9e, from where it is perceived that the component always experiences an increase with AoA. This increase is steeper as J grows. It is also noticeable that C_{Twing} is very small at low angles, where it grows slightly and then recedes with J. As AoA grows, the amplitude of this "hump' shaped curve increases, until eventually at high angles, C_{Twing} will no longer decrease with J.

For all the propellers tested, it can be seen that C_T decreases with J growth, at low angles, and for the lower pitch propellers, the windmill state ($C_T = 0$), happens at lower J values, according to Figs. 6.6a, 6.7a, 6.8a, 6.9a. As AoA increases, the value of J for windmilling is higher, until eventually, at a high AoA, C_T starts growing with J, and no windmilling occurs. The angle for the behavioural inversion is not clearly distinguishable, from these surfaces plots.

6.5 The Transition Angle from Propeller to Wing Behaviour

In the previous section, the behaviour of C_T surfaces was inspected for the propellers tested, and although it was clearly evident that an inversion of C_T behaviour with respect to J occurred at high angles ($AoA \approx 60^o$ or higher) in the test cases investigated here, the exact angle region was indeterminate, in each case. In this section we will investigate, through the interpolated curve functions of each propeller, at which angle this behavioural inversion happens. An explanation for the phenomenon, in light of the new developed theory, will be presented.

From the C_T surface fits of each propeller, plots of C_T vs AoA are produced, for selected J values. This is presented in Figs. 6.10a, 6.10c, 6.10e, 6.10g. Also, plots of

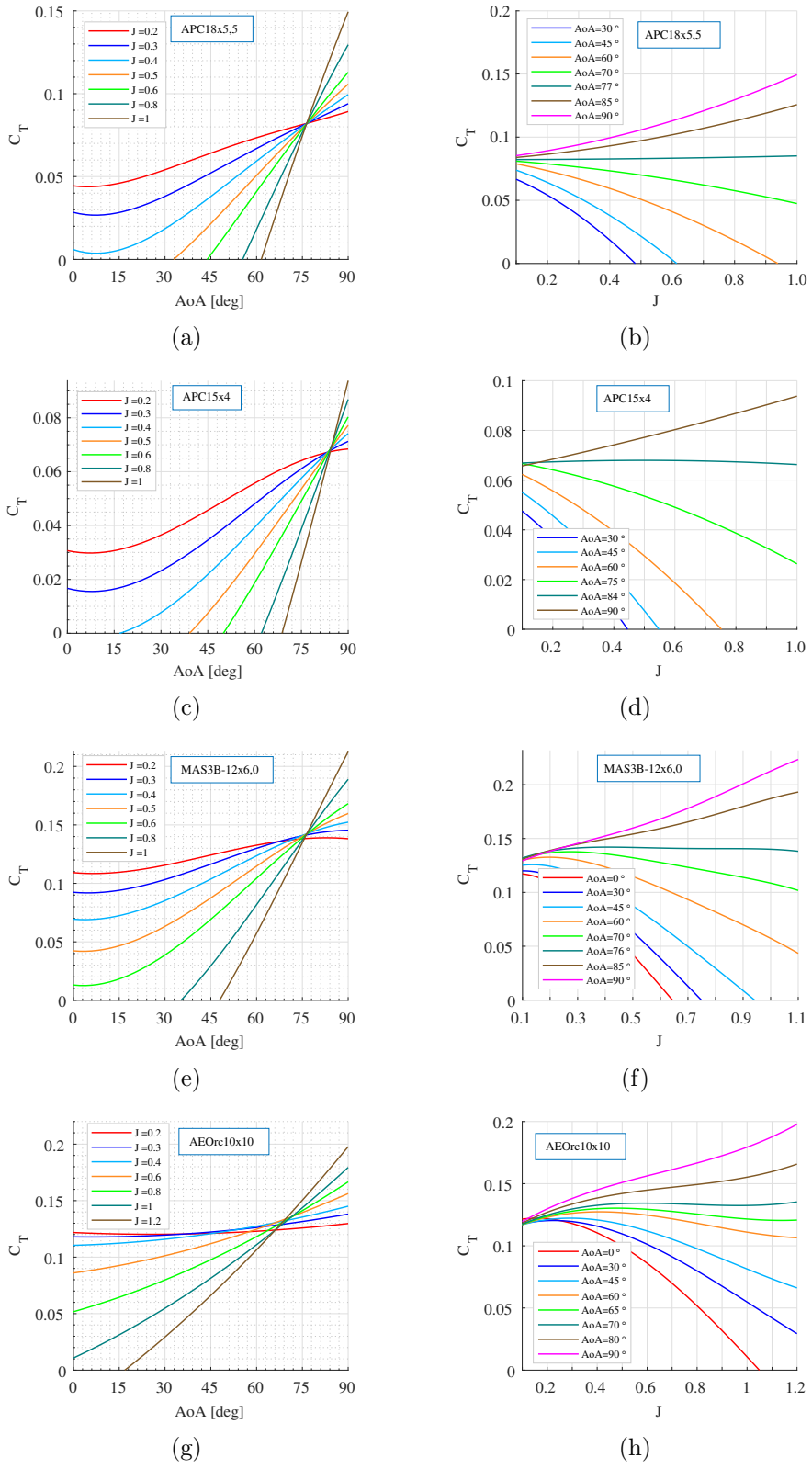


Figure 6.10: Propellers thrust coefficients: (a),(b) APC 18x5.5", (c),(d) APC 15x4", (e),(f) MAS 3B-12x6", (g),(h) AEOrc 10x10"

C_T *vs* J for several chosen angles, are presented in Figs. 6.10b, 6.10d, 6.10f, 6.10h.

From Figs. 6.10a, 6.10c, 6.10e, 6.10g, it is seen that as J increases, C_T is reduced at low angles, and the slope $\partial C_T/\partial \alpha_p$ grows with J. This slope increase will eventually cause the inversion of C_T behaviour, to grow with J. This occurs at $AoA \approx 77^o$ for the propeller APC 18x5.5", at $AoA \approx 84^o$, for the APC 15x4", at $AoA \approx 76^o$ for the MAS 3B-12x6", and at $AoA \approx 70^o$ for AEOrc 10x10".

The same effect can be analysed through Figs. 6.10b, 6.10d, 6.10f, 6.10h, as C_T is plotted against J, for several AoA. At the inversion angles mentioned, for all the propellers, the sensitivity inverts, and so $\partial C_T/\partial J \approx 0$, along the J abscissa.

Now, recalling the new developed theory, it is mentioned that the inversion of C_T behaviour could only occur at angles high enough, where the wing component of thrust is relevant, and most importantly, where its influence, $(\partial C_{T\,wing}/\partial J)$, is. As T_{wing}, or $C_{T\,wing}$, begins to show at angles $AoA > 30^o$, growing to become increasingly important at around $AoA \gtrapprox 60^o$, then it is also to be expected that $\partial C_{T\,wing}/\partial J$ will become influential at angles around the same region. The results for the propellers tested here (all having $\beta_{0.75} \leq 23^o$) showed the inversion of thrust behaviour at angles $AoA > 60^o$, which is in agreement with the experiment results obtained in [48, 49, 94, 95], and in McLemore and Cannon [26] for propellers with blade pitch angles $\beta_{0.75} < 25^o$. Yaggy and Rogallo [28] found the inversion happening at $AoA \geq 60^o$, for propellers with blade angles lower than 25^o, whereas for the higher $\beta_{0.75} = 40^o$, the inversion was around $AoA \approx 50^o$. A further analysis of this inversion is now offered, in light of the new theory, under the scope of the thrust components $C_{T_{axial}}$, and $C_{T_{wing}}$ influence.

Figure 6.11 shows several cross section plots of the C_T, $C_{T\,axial}$, and $C_{T\,wing}$ surfaces (Fig. 6.6b) of propeller APC 18x5.5". The sections are taken at different angles, including the angle $AoA = 77^o$, seen in Fig. 6.10b, where the behavioural inversion of C_T *vs* J occurs. This angle of thrust behavioural inversion is hereafter denominated as AoA_{inv}. An inspection of the thrust coefficients reveals, that:

- At the angle of thrust behavioural inversion AoA_{inv}, C_T is approximately con-

stant with J, and so $\partial C_T/\partial J \approx 0$ (here the very high J condition, where retreating blade stall happens, is not considered). Above the referred angle, $\partial C_T/\partial J > 0$. Below AoA_{inv}, C_T always drops with J, implying on $\partial C_T/\partial J < 0$. The decrease is more intense below 60^o.

- As in Fig. 6.6d, at low angles, C_{Taxial} is predominant over C_{Twing}, and its behaviour is responsible for practically all of C_T behaviour. Also, at $AoA < 60^o$, C_{Taxial} diminishes rapidly with J, to reach zero (windmilling). Between 60^o, and AoA_{inv}, the reduction with J is milder, but eventually, windmilling will still happen. At high AoA, past the angle of C_T behavioural inversion AoA_{inv}, its reduction is asymptotic as J grows, and C_{Taxial} never completely vanishes.

- As in Fig. 6.6e, C_{Twing} can be noticed to be negligible at low angles. It grows and then recedes with J down to zero at windmill state, featuring a "hump" shaped curve that increases in amplitude with AoA, more noticeably above 60^o.

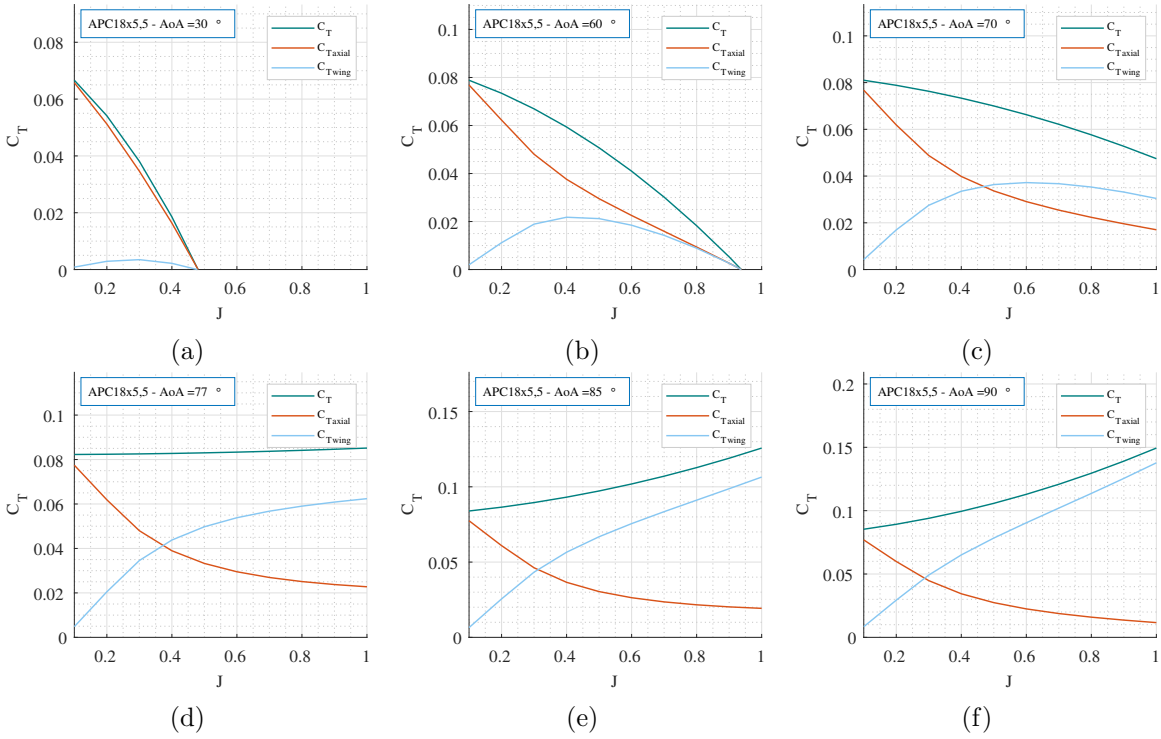


Figure 6.11: Propeller APC 18x5.5" C_T surfaces cross sections, at assorted AoA values: (a) $AoA = 30^o$, (b) $AoA = 60^o$, (c) $AoA = 70^o$, (d) $AoA = 77^o$, (e) $AoA = 85^o$, (f) $AoA = 90^o$

Above AoA_{inv}, it does not recede any more, but keeps increasing with J.

- C_{Twing} overcomes C_{Taxial} only at angles higher than 60^o, as seen also from the surfaces in Fig. 6.6c. Here, in Fig. 6.11, these points can be clearly identified, for each angle. For instance, at $AoA = 70^o$, in Fig. 6.11c, the two components C_{Taxial}, and C_{Twing} equalize at $J \approx 0.47$. As J grows, C_{Twing} overcomes C_{Taxial}, but, as at this angle the propeller still reaches windmill state, eventually, both components fall to zero at $w/V = 0$, which corresponds to the ordinate of Fig. 6.4a, where $C_{Taxial} = 0$, and $C_{Twing} \rightarrow C_{Taxial}(T_{wing}/T_{axial})$, also vanishes. By checking $J \approx 0.47$ against w/V, from the fits w/V *vs* J in Fig. 6.5b, it is seen that it corresponds to $w/V \approx 0.2$. Now, the inspection of Fig. 6.4a for the ratio T_{wing}/T_{axial}, confirms that the value of w/V must be ≈ 0.2 for the ratio to be 1, at $AoA = 70^o$, (see Fig. 6.4), which also should be valid for any propeller, as per Eq. (3.15).

- At the angle of C_T behavioural inversion ($AoA_{inv} = 77^o$), it can be seen in Fig. 6.11d that C_{Taxial} and C_{Twing} curves, evolve in an exact opposite way with J, and so, the sensitivities of the two components cancel out along the J abscissa. It is fair to assume then, that at the inversion angle AoA_{inv}, for any J, $\partial C_{Taxial}/\partial J = -\partial C_{Twing}/\partial J$.

- For any angle at very low J, the propeller of course is close to static condition, and C_{Taxial} is the sole component of thrust. At angles higher than AoA_{inv}, C_{Taxial} quickly loses ground to C_{Twing}, as J picks up. Soon, $\partial C_{Twing}/\partial J > |\partial C_{Taxial}/\partial J|$, which will cause $\partial C_T/\partial J > 0$. As J continues to grow, then C_{Twing} also becomes much larger than C_{Taxial} (see Figs. 6.11e, 6.11f).

Similar observations, as above, can be made from Figs. 6.12, 6.13, 6.14 for the remaining propellers tested, and the results interpretation must hold in general. Important emphasis is on the different particular C_T behavioural inversion angles of each propeller, being, $AoA_{inv} \approx 84^o$ for the APC 15x4", $AoA_{inv} \approx 76^o$ for the MAS 3B-12x6", and $AoA_{inv} \approx 70^o$ for AEOrc 10x10". Apparently from

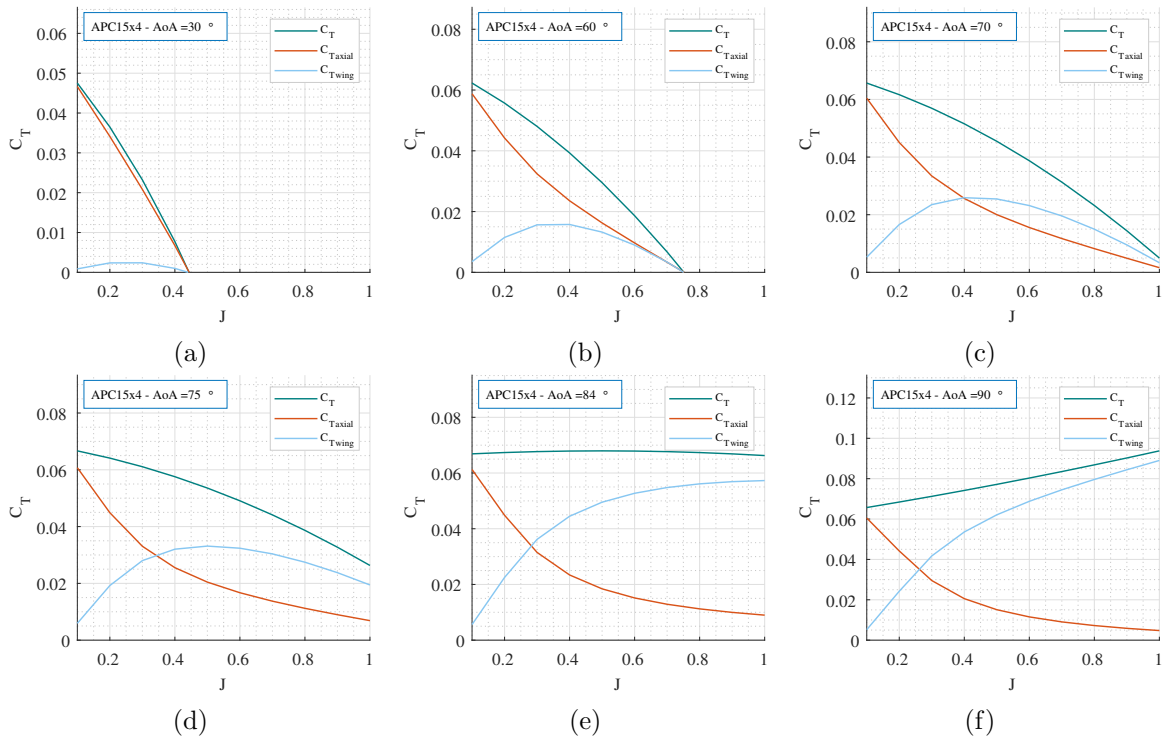


Figure 6.12: Propeller APC 15x4" C_T surfaces cross sections, at assorted AoA values: (a) $AoA = 30^o$, (b) $AoA = 60^o$, (c) $AoA = 70^o$, (d) $AoA = 75^o$, (e) $AoA = 84^o$, (f) $AoA = 90^o$

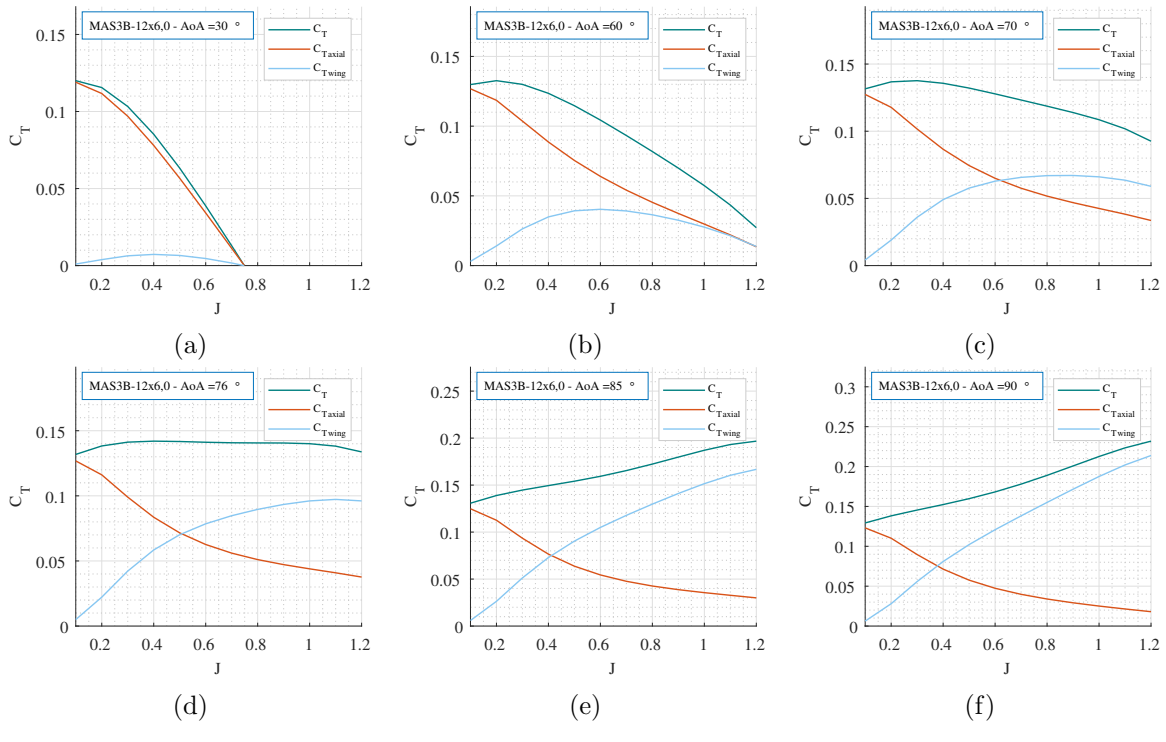


Figure 6.13: Propeller MAS 3B-12x6" C_T surfaces cross sections, at assorted AoA values: (a) $AoA = 30^o$, (b) $AoA = 60^o$, (c) $AoA = 70^o$, (d) $AoA = 76^o$, (e) $AoA = 85^o$, (f) $AoA = 90^o$

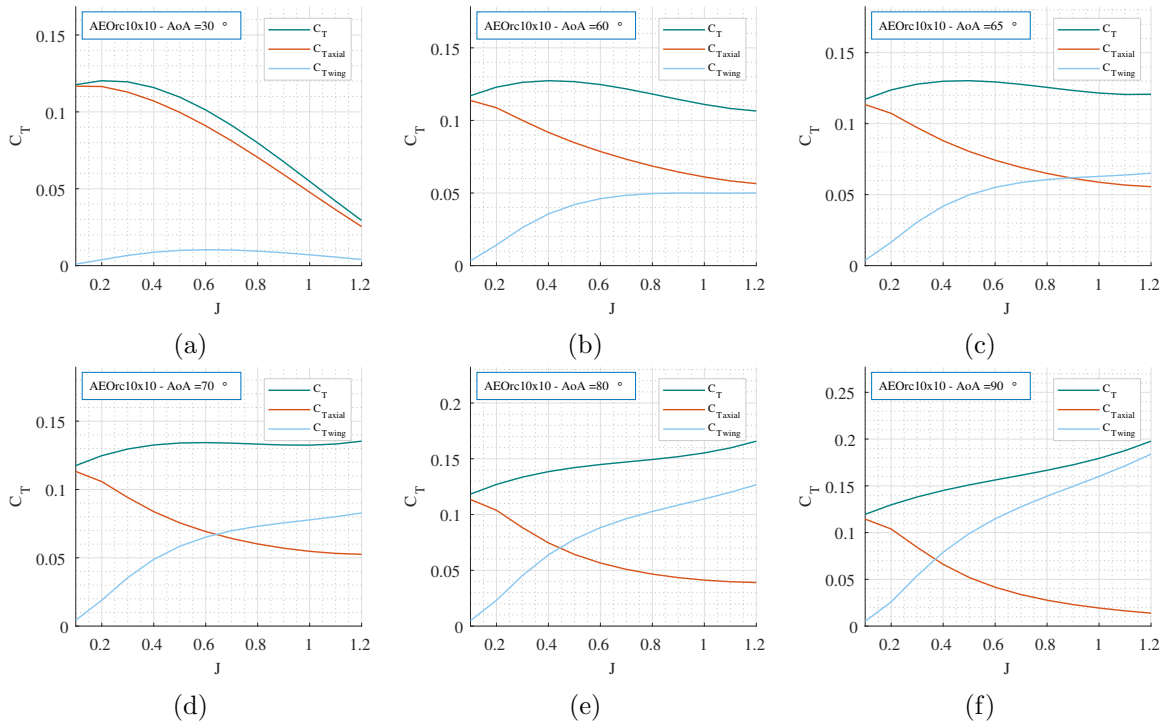


Figure 6.14: Propeller AEOrc 10x10" C_T surfaces cross sections, at assorted AoA values: (a) $AoA = 30^o$, (b) $AoA = 60^o$, (c) $AoA = 65^o$, (d) $AoA = 70^o$, (e) $AoA = 80^o$, (f) $AoA = 90^o$

the results here, AoA_{inv} seems to happen at lower angles of incidence AoA, for propellers with higher blade pitch angles. This is the case also in the results from McLemore & Cannon [26], and Yaggy & Rogallo [28], where the inversion angle happens at AoA even lower than 60^o, for exceedingly high blade pitch angles.

It has been found that propellers at high incidence, under translational velocity, behave much as wings ([31, p.319], [42, p.127],[48, p.124]). For all the propellers tested in this study, it has been shown that for angles of incidence higher than the C_T behavioural inversion angle, propellers behave in a very different mode than at angles lower than AoA_{inv}, with respect to the free-stream velocity. First, for the wing component of thrust, being the far largest responsible for the thrust, and second, for the sensitivity of thrust becoming positive, causing thrust to increase with the free-stream velocity, as occurs with wings lift. In that sense, the propeller is behaving much like a wing, above the referred angle, and under translational speed. Therefore,

under these considerations, the author finds plausible to define AoA_{inv}, as the angle at which propellers transition to wing behaviour.

6.6 Concluding Remarks

In this chapter we have confirmed that the relation of C_{Twing}/C_{Taxial} vs w/V at angles of incidence must be the same for any propeller, and that the peculiarities of each propeller can be noticed through the different surfaces of C_T vs J, and AoA, which arise as each propeller presents its respective w/V vs J curve.

Also, we addressed the suggestion for further work in the concluding remarks of chapter 5 , "... the ratio T_{wing}/T_{axial} must follow the theory and the inversion of thrust behavior should happen around the region where T_{wing} becomes relevant, and $(\partial T_{wing}/\partial V)$ overcomes $(\partial T_{axial}/\partial V)$, which happens at high angles and at high speeds $(w/V < 0.6)$". This verification has been done in this chapter, to show that indeed for the propellers tested, the inversion of thrust behaviour can only happen at increasing J, and at angles of incidence high enough, so that the influence of the wing component is paramount (generally at $AoA \geq 60^o$, for blade angles $\beta_{0.75} < 25^o$, found to be the case also in the literature cited previously). More importantly, the behavioural inversion only happens after $\partial C_{Twing}/\partial J$ overcomes $|-\partial C_{Taxial}/\partial J|$. The C_T behavioural inversion angle AoA_{inv}, where, $\partial C_T/\partial J \approx 0 \approx \partial C_{Twing}/\partial J + \partial C_{Taxial}/\partial J$ determines when every propeller transitions to wing behaviour.

Chapter 7

Conclusions, & Future Work

7.1 Conclusions

In this work, a series of experimental tests were conducted to investigate the aerodynamic thrust performance of propellers operating at angles of incidence (AoA), ranging from 0° to 90°. This was done in a closed-loop wind tunnel at the University of Canterbury. An alternative approach to analyze propeller's thrust performance is proposed. It is based on the classical Momentum Theory, where a theoretical entrainment factor e is defined to account for the mass flow rate, through an enhanced area of the rotor disk, to equalize Glauert's Hypothesis mass flow definition. The factor e is found to depend on the angle ϵ, defined as the angle between the air velocity vector at the disk, $\mathbf{V_{disk}}$ and the thrust vector $\mathbf{T}$. It is shown mathematically that the thrust T consists of two components; one is the axial component T_{axial}, and the other is the wing lift equivalent component T_{wing}. It is shown in the research, that the behaviour of propellers thrust can be explained through the influence of the two components. Initially, experiments were conducted on a 6 inch diameter, 4.5 inch/revolution pitch, 2-blade propeller, having a blade pitch angle $\beta_{0.75} \approx 17.6^o$. Subsequently, more experiments were performed on four different fixed-pitch propellers, with blade pitch angles up to $\beta_{0.75} = 23^o$, to further investigate the new theoretical development. The advance ratios J in the tests ranged from 0 up to 1.2. The main findings of the research are summarized below:

- For all tested propellers, below a certain high angle of incidence, thrust T was

found to be reduced with V, $(\partial T/\partial V < 0)$, for a given RPM, or in general terms, C_T is reduced with the advance ratio J, $(\partial C_T/\partial J < 0)$. For $AoA < 60^o$, this reduction is more intense. As AoA increases past 60^o, the decrease of C_T, with J becomes milder, until eventually, above that referred high angle of incidence, and under increasing airspeed, there is a change in behaviour, and the propeller sensitivity to airspeed inverts to become positive, $(\partial C_T/\partial J > 0)$. This referred angle, where the thrust behavioural inversion takes place, is defined as AoA_{inv}. The results indicated that AoA_{inv} should decrease for increasing blade pitch. This was also found in some limited results of the literature.

- T_{axial} behaves similarly to a propeller operating at no incidence, under an axial airflow of magnitude $V \cos \alpha_p$. Therefore, for a given RPM, it is decreased with increasing airspeed $(\partial T_{axial}/\partial V < 0)$, or in terms of coefficients and advance ratio, $(\partial C_{Taxial}/\partial J < 0)$. At low angles, C_{Taxial} is predominant over C_{Twing}, being C_{Taxial} responsible for practically all of C_T behaviour in that region. Also, at $AoA < 60^o$, C_{Taxial} diminishes rapidly with J, to reach zero (windmilling). Between 60^o, and AoA_{inv}, the reduction with J is milder, but windmilling will still happen. At high AoA, past the C_T behavioural inversion angle AoA_{inv}, its reduction is asymptotic as J grows, and C_{Taxial} never completely vanishes. C_{Taxial} is shown to be not so much sensitive to AoA, at low J. As J grows, C_{Taxial} begins to vary with AoA, decreasing at high AoA, and J.

- T_{wing} provides the equivalent lift of an elliptic wing, under velocity $V \sin \alpha_p$ of magnitude with a variable area equal to S_{disk} multiplied by a factor WF. This factor ranges from zero at $AoA = 0^o$ to $WF \to 1$, as AoA approaches 90° at high airspeed V. T_{wing} was found to be negligible at low angles. It increases with AoA, to be meaningful at high angles. Also the sensitivity slope of T_{wing} to AoA increases with V. In terms of coefficients, below AoA_{inv}, C_{Twing} grows and then recedes with J, down to zero, at the windmill state, featuring a "hump" shaped curve that increases in amplitude with AoA, more noticeably above 60^o, where it contributes to attenuate the intensity of C_T reduction with J. Eventually, above AoA_{inv}, it does not recede any more, but continues in an ever increasing

response to J, being the main responsible for the C_T behavioural inversion. It is important to mention again that, the analysis does not consider extremely high J conditions where flow separation and blade stall may occur.

- The theory shows that T_{wing} surpasses T_{axial} at $AoA > 60^o$, and at high speeds ($w/V < 0.57735$). Also, the ratio T_{wing}/T_{axial} vs w/V is independent of the blade geometry and the propeller design. These assumptions were made in deriving the Momentum Theory, and verified in the tests of the four propellers. The influence of T_{wing} on T overcomes that of T_{axial} at high angles, and speed. The growing sensitivity $\partial C_{Twing}/\partial J$ with AoA also becomes ever more important as AoA grows towards 60^o and over, to eventually overtake $-\partial C_{Taxial}/\partial J$, at the angle of thrust behavioural inversion, AoA_{inv}.

- The increase of T with AoA is due, to a great extent, to the wing component. T_{wing} peaks to compose most of the thrust at very high speed, and high AoA. In this case, the proposed theory is shown to be consistent with Glauert's Hypothesis, reverting to the wing lift formula used for helicopter rotors in fast forward flight, $T \rightarrow 2\rho V w S_{disk}$.

- At $AoA \rightarrow 90°$, T_{axial} converges to the static thrust formula, being always present in the thrust composition even in forward flight. At hovering condition, when T_{wing} vanishes, then T_{axial} becomes the sole contributor of T, in which case the theory is proven to be consistent with the traditional formula $T = 2\rho S_{disk} w^2$.

- The proposed theory is also shown to agree completely with the classic Momentum Theory at AoA=0°. The development presents an alternative simplified formula for estimating T at angles of incidence, based on data acquired at $AoA = 0^o$, (T or C_T, and V, RPM or J). The formula is intended for up to intermediate advance ratios J, far from the windmill state. It showed good agreement with the tests data for the propeller tested in the 1st tests campaign, in the region where T_{axial} was relatively constant with AoA.

Propellers at high incidence, under translational velocities have been known to behave much as wings. For all the propellers tested, it has been shown for the

underlying conditions of this study, that for angles of incidence higher than the C_T behavioural inversion angle, propellers behave in a very different mode than at angles lower than AoA_{inv}, with respect to the free-stream velocity. First, for the wing component of thrust, being the far largest responsible for the thrust, and second, for the sensitivity of thrust becoming positive, causing thrust to increase with the free-stream velocity, as occurs with wings lift. In that sense, the propeller is behaving much like a wing, above the referred angle, and under translational speeds. Therefore, under these considerations, the author finds plausible to define AoA_{inv}, as the angle at which propellers transition to wing behaviour.

It is important to stress again that the tests range in this work do not reach far up to very high advance ratios J, where thrust reduction could again occur at high AoA, due to reverse flow separation, and stall on some sections of the retreating blade, and due to possible stall conditions also in the advancing blade, in the case of propellers with exceedingly high blade pitch angles. The author believes that the condition of insensitivity to J at the inversion, defining AoA_{inv}, should be valid below those extreme ranges. Within these assumptions and those of the momentum theory, the analysis and conclusions of this work are presumed to be valid for propellers in general.

Lastly, a summarizing conclusion that can be reached from this research is that one could assume the thrust of a propeller at incidence to be interpreted as the thrust of a propeller in axial flow condition, under incoming speed of $V \cos \alpha_p$, with a thrust addition equivalent to the lift produced by an elliptic wing of area $S_{disk}WF$, under incoming speed $V \sin \alpha_p$, where both components share a common induced speed w.

7.2 Final Comments and Recommendations for Future Work

The work done in this research focused on the thrust analysis of propellers at incidence. Although, side forces and moments are of great concern when it comes to propellers performance analysis, the novelty of this work is related to the thrust investigation, and so, presenting results that would not introduce any new knowledge was

not done. Nevertheless, these forces and moments readings were collected, mainly for efficiency analysis that requires torque inputs. The author believed that the moment readings from the balance would suffice for the calculations, however it turned out that the results acquired were not consistent and efficiency figures were low, and not reliable in most cases. It is suspected that these discrepancies were caused by the slipstream impinging on the holding rig, which would distort moments and forces readings. Ideally, the sensors, including a torque transducer, should have been placed right aft of the propeller to eliminate this problem. Due to the limiting time for preparation, and to carry out new tests, unfortunately this was not done. Therefore it remains as a recommendation for future work, where it would be interesting to evaluate the increase of propeller's efficiency range with increasing AoA, found in the general literature, under the scope of the theory, according to Section 3.6 in Chapter 3. The author believes that this range widening could be caused by the wing component of thrust.

A second recommendation is to evaluate the theory applied for propellers at very high blade pitch angles.

Finally, another potential useful study would be to compare the results from the theory, to an analysis under the scope of the blade elements, where the velocity vectors would include the velocity at the disk at the angle ϵ, and possibly, decompose the resultant vectors in axial and in-plane components to try and isolate the axial and wing components effects. If at all possible, one could then separate the propellers performance, under the scope of the blades elements, into a propeller operating at axial condition, with the addition of a wing lift, of some specific area.

Appendix A: Precision Errors of the 2nd Campaign Tests

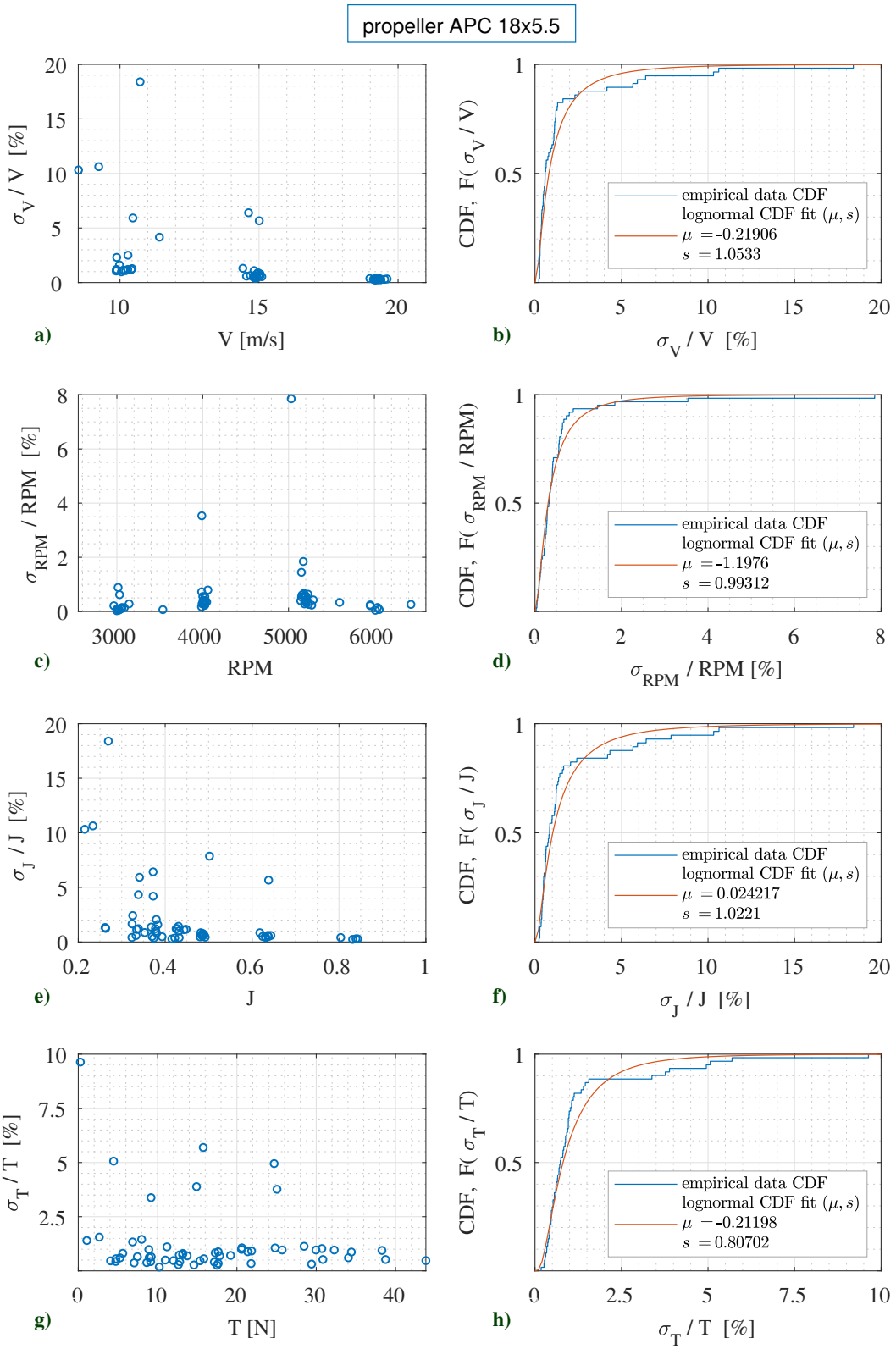


Figure A.1: Measurement precision errors for propeller APC-18x4.5": (a) in V, (c) in RPM, (e) in J, (g) in T, and (b), (d), (f), (h) are their respective Cumulative Distribution Functions.

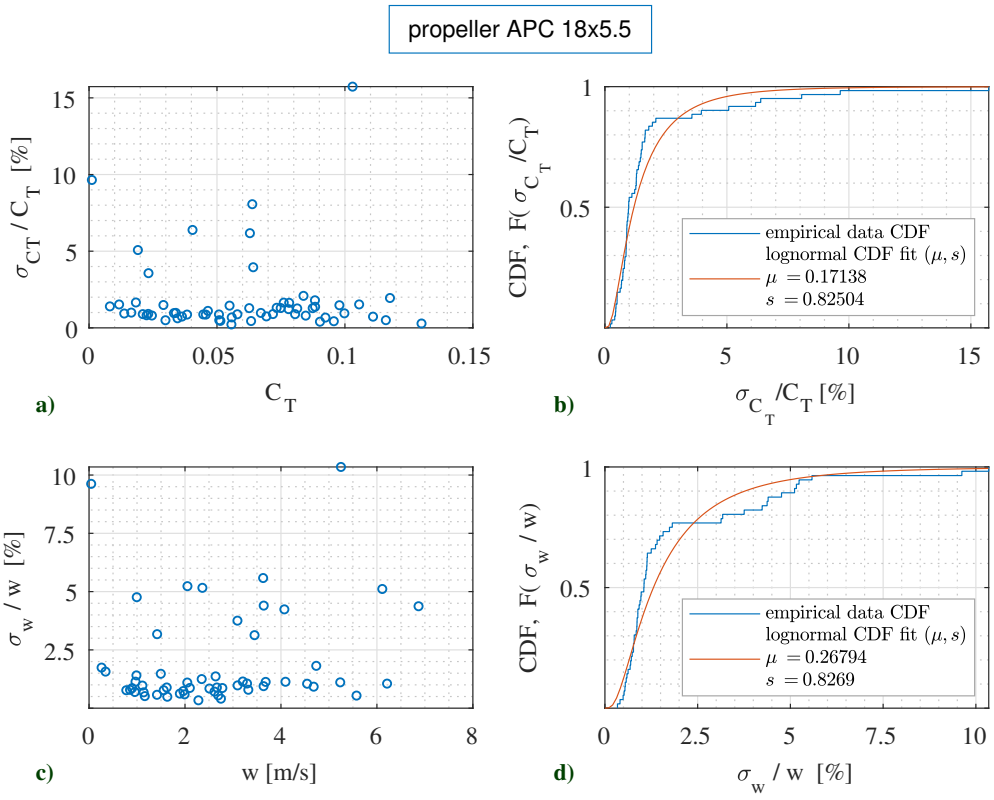


Figure A.2: Measurement precision errors for propeller APC-18x4.5": (a) in C_T, (c) in w, and (b), (d) are their respective Cumulative Distribution Functions.

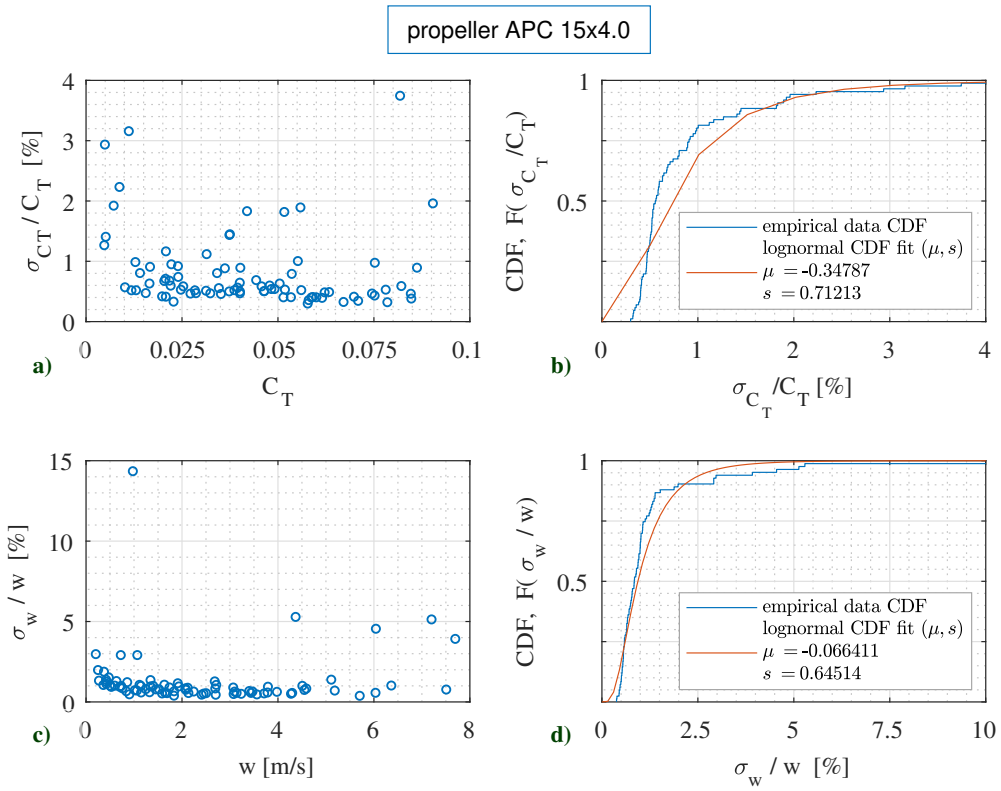


Figure A.3: Measurement precision errors for propeller APC-15x4": (a) in C_T, (c) in w, and (b), (d) are their respective Cumulative Distribution Functions.

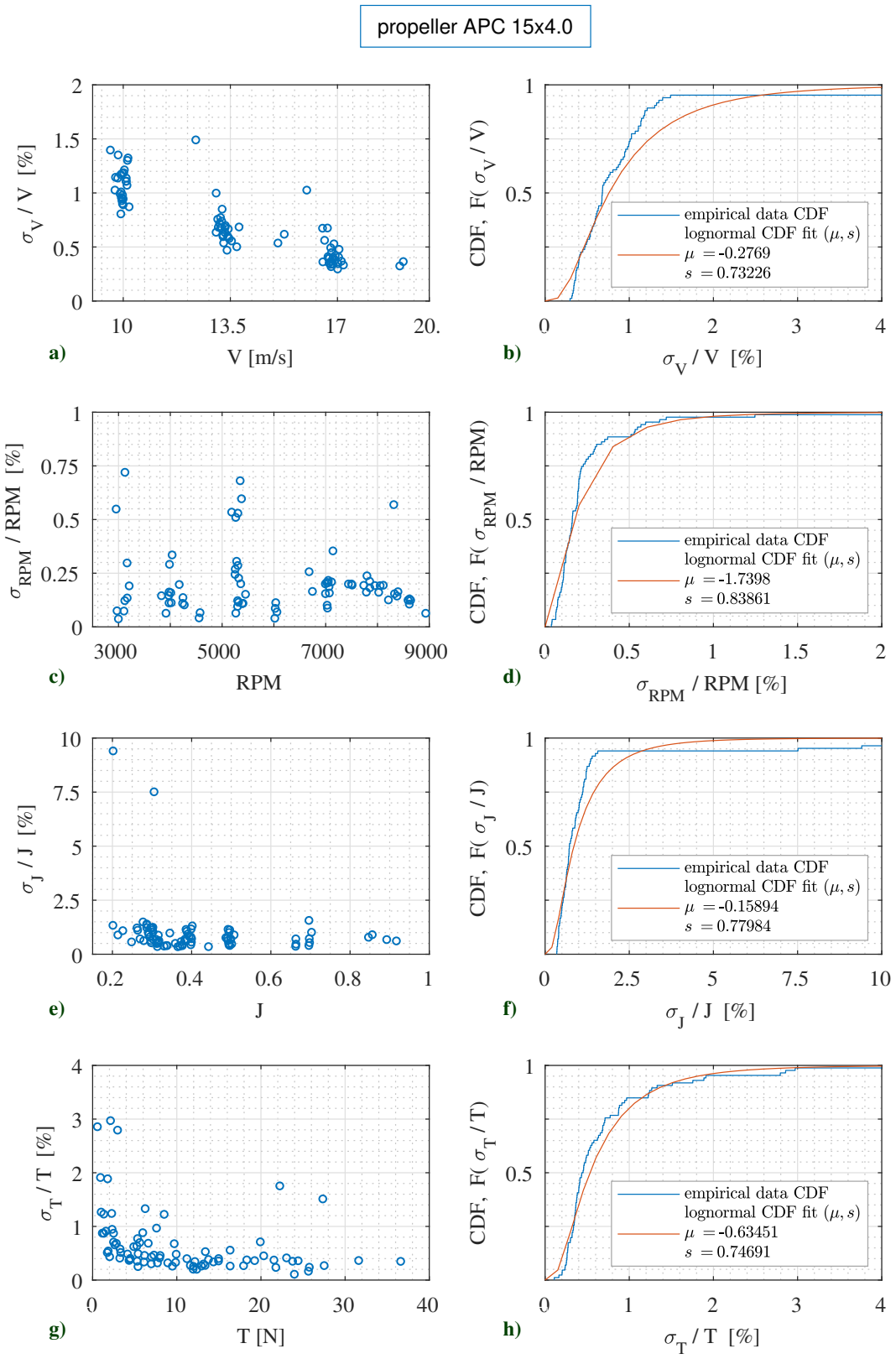


Figure A.4: Measurement precision errors for propeller APC-15x4": (a) in V, (c) in RPM, (e) in J, (g) in T, and (b), (d), (f), (h) are their respective Cumulative Distribution Functions.

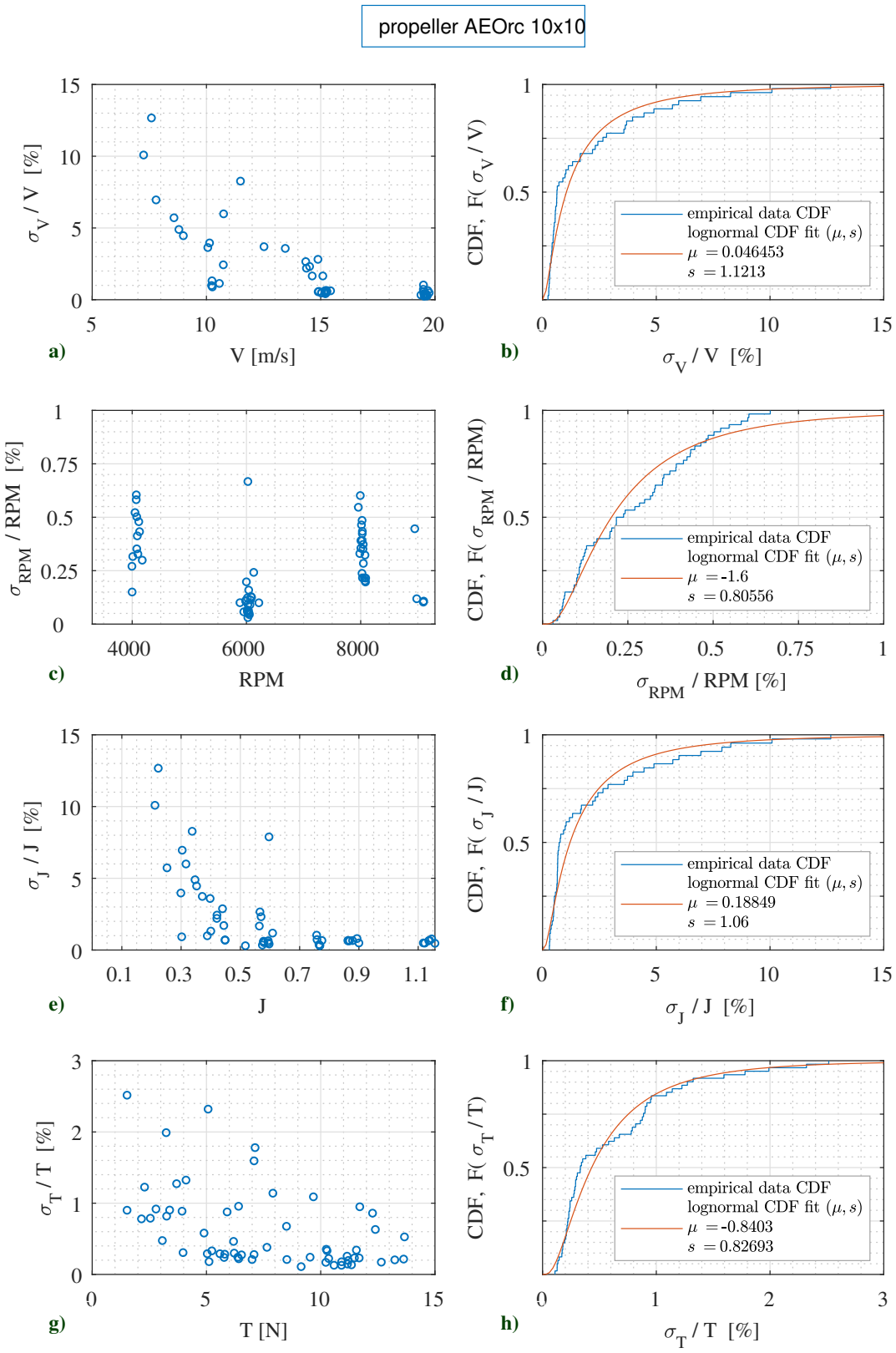


Figure A.5: Measurement precision errors for propeller AEOrc-10x10": (a) in V, (c) in RPM, (e) in J, (g) in T, and (b), (d), (f), (h) are their respective Cumulative Distribution Functions.

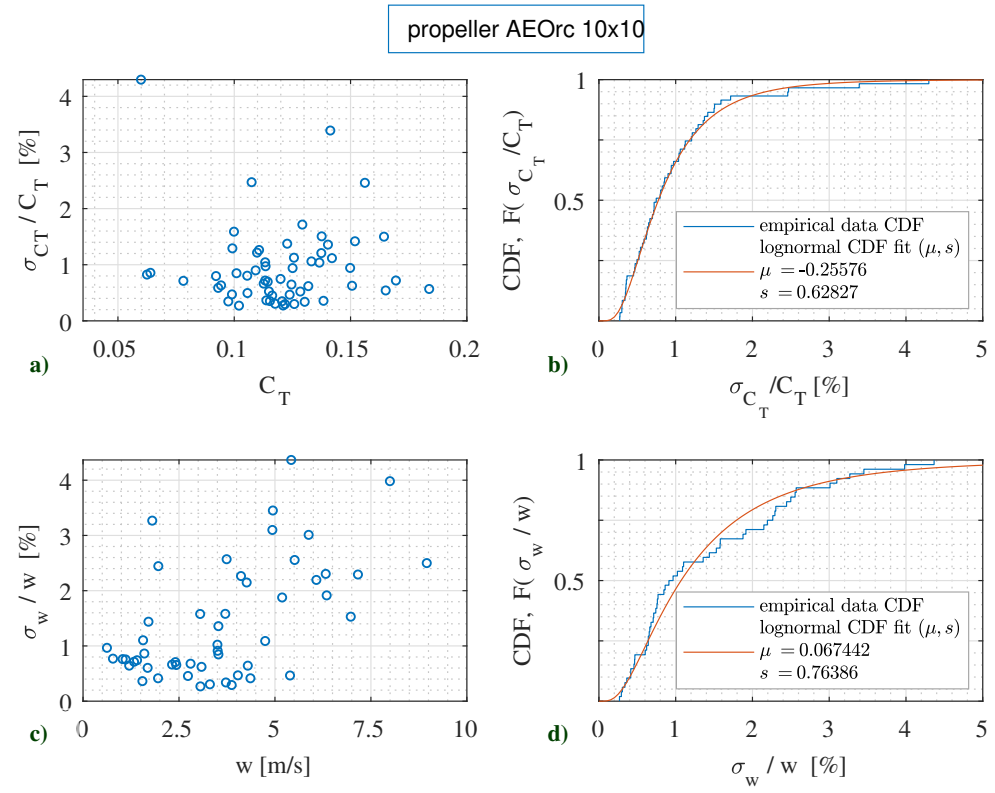


Figure A.6: Measurement precision errors for propeller AEOrc 10x10": (a) in C_T, (c) in w, and (b), (d) are their respective Cumulative Distribution Functions.

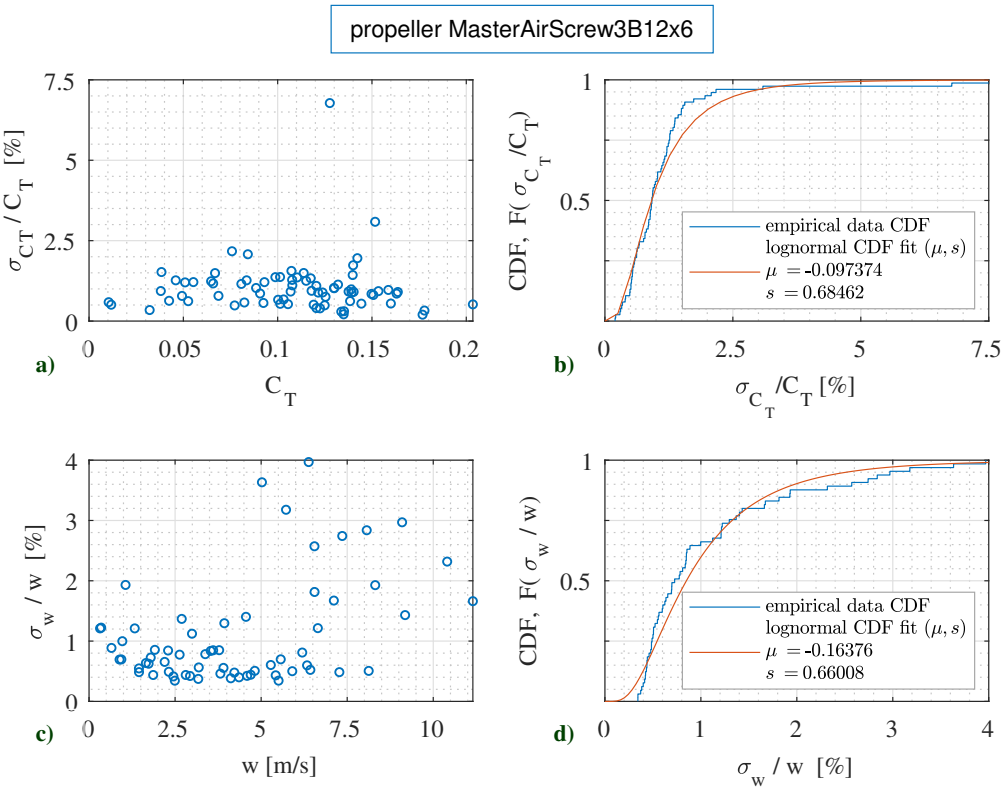


Figure A.7: Measurement precision errors for propeller MAS 3B-12x6": (a) in C_T, (c) in w, and (b), (d) are their respective Cumulative Distribution Functions.

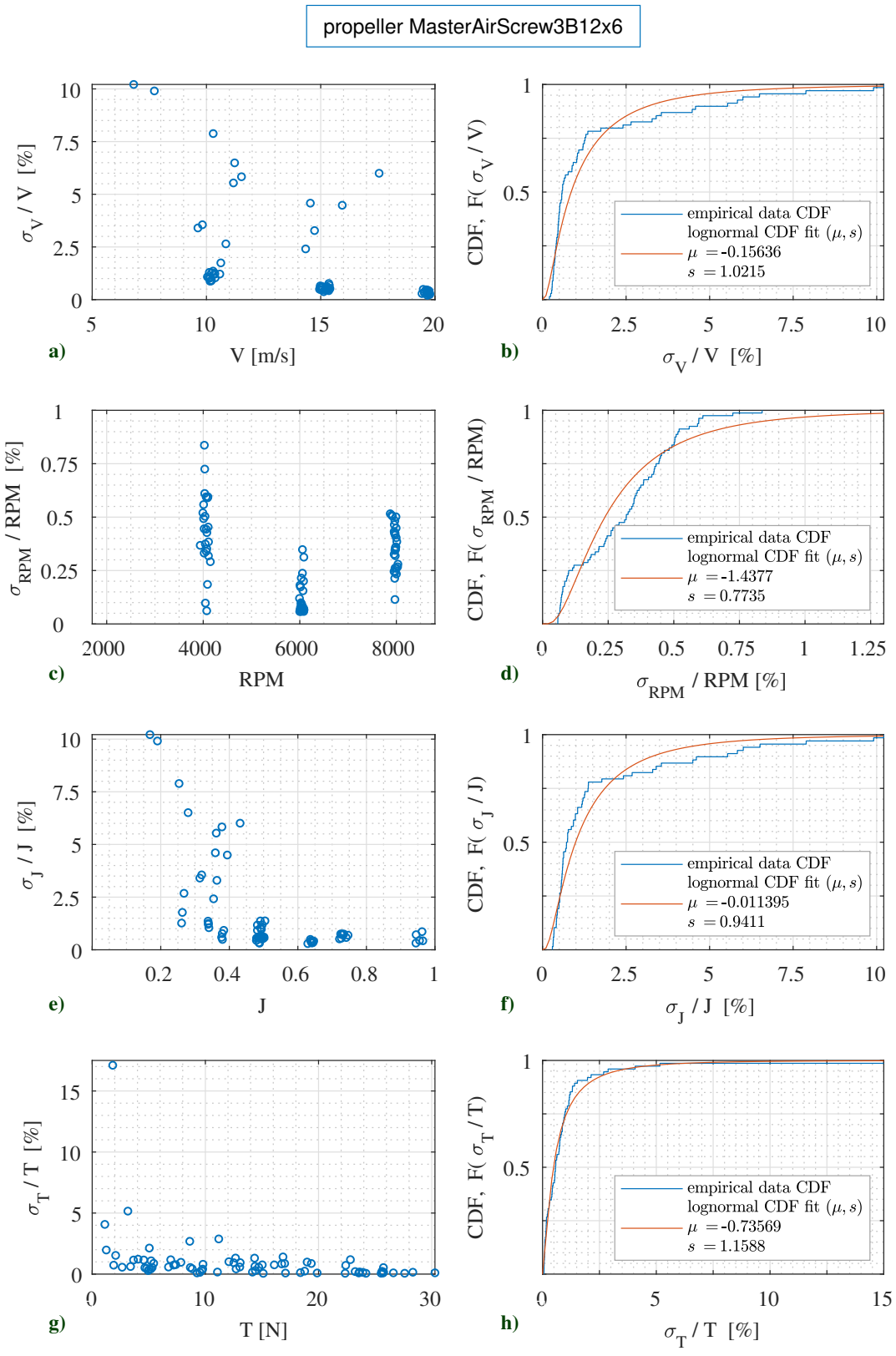


Figure A.8: Measurement precision errors for propeller MAS 3B-12x6": (a) in V, (c) in RPM, (e) in J, (g) in T, and (b), (d), (f), (h) are their respective Cumulative Distribution Functions.

Appendix B: Description of the Data Processing Scripts

B.1 Script to read and compile rig calibration tests data

The resulting data from the rig resistance tests, for every AoA, and V condition, are stored in .txt files, in a folder named "Calibration Rig". A screenshot, taken of the "Calibration Rig" folder, with several .txt data files with different AoA and V conditions, is presented Fig. B.1.

The data files acquired, are read by a script named "Read_CalibrationRIG_-TestFiles_Script.m", (see Appendix C.1). The script opens every file, for every test condition inside the specific folder, reads the data, and calculates the average and standard deviation of all forces. For every condition, the averages and standard deviations estimated are stored in one line of an array. After all files, in the folder, are compiled, the script will have created a table, that is stored in a file containing the names "CALIBRATION_RIG_ .... _ResultAcquired.mat".

The script also creates 6 fits for forces and moments, using the data from the newly created table, that also allows for the creation of surface plots of every force, against AoA and V. The fits are stored in a file containing the names "CalibRIG_-.... FITS.mat". The plots are stored in a file containing the names "CalibRIG_ ..... _graphs.fig". The graphs are then visualized for inspection of the results quality, and whether some outliers need to be excluded. The fits are of the type "thinplateinterpolation" option, from MATLAB. Their equations, in the form $f(AoA, V)$, will be used to net the forces measured from the propellers tests. Figure B.2 depicts the calibration rig forces acquired for the second tests campaign using the respective rig

assembly as described in Chapter 6. It shows the moments, and the relevant forces F_x and F_y, used in the calculations of T, or N_p, according to Eqs. (6.1a), and (6.1b).

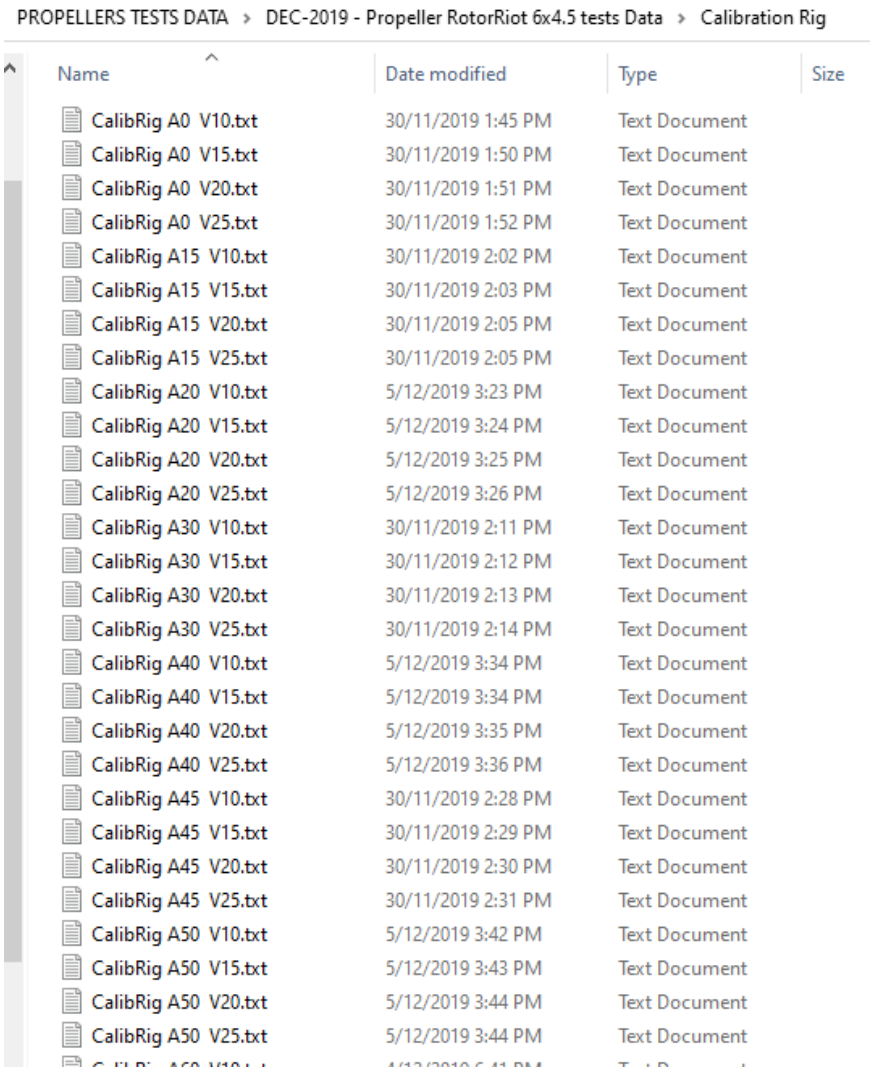
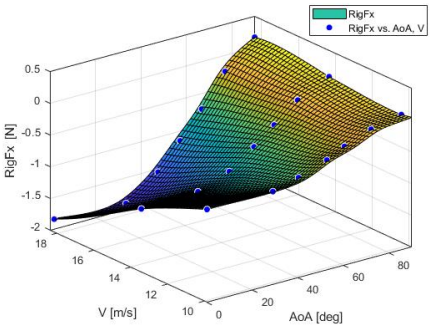
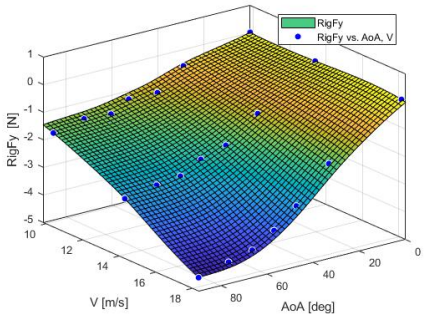


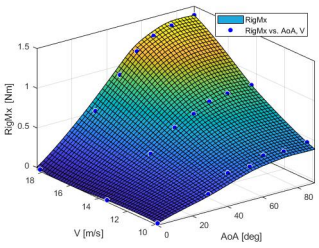
Figure B.1: Screenshot of the folder containing .txt files for rig calibration



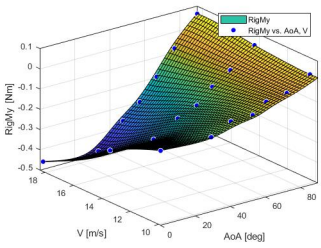
(a) F_x



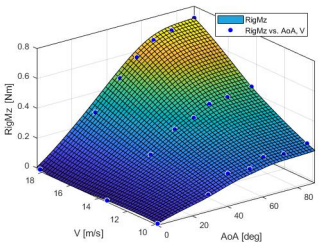
(b) F_y



(c) M_x



(d) M_y



(e) M_z

Figure B.2: Resisting forces of the rig used in the 2nd tests campaign, measured without the propeller

B.2 Script to read and compile propellers tests data

The propeller tests generate files in .txt format for the forces recorded, and in .xlxs format, for power, and RPM captured. These files are stored in a folder after the propeller name, and under a main folder "Propellers Tests Data". A screenshot, showing the aspect of the folder that contains .txt and .xlxs format files with propellers tests data, for different AoA, V, and RPM conditions, is presented in Fig. B.3.

A script to open, read, and compile the propeller tests data files, has been created and named "Read_Prop_Test_DATA.m", (see Appendix C.2). Similarly to the previous script, .txt data files are read, but this script reads also .xlsx format files, containing electric power data, and RPM values. In that sense, the script opens and reads every pair of files, for every test condition, inside the specific folder "... Propellers Tests data/ *file_name_after_propeller*". It then, calculates the average, and

standard deviations of all forces, moments and power measurements. Every test condition will then constitute a line of an array created to store the data. After all files in the folder are compiled, the script will have created a table named "*name_of_propeller*_ResultAcquired_Table", that is stored in a file named "*name_of_propeller*_ResultAcquired.mat".

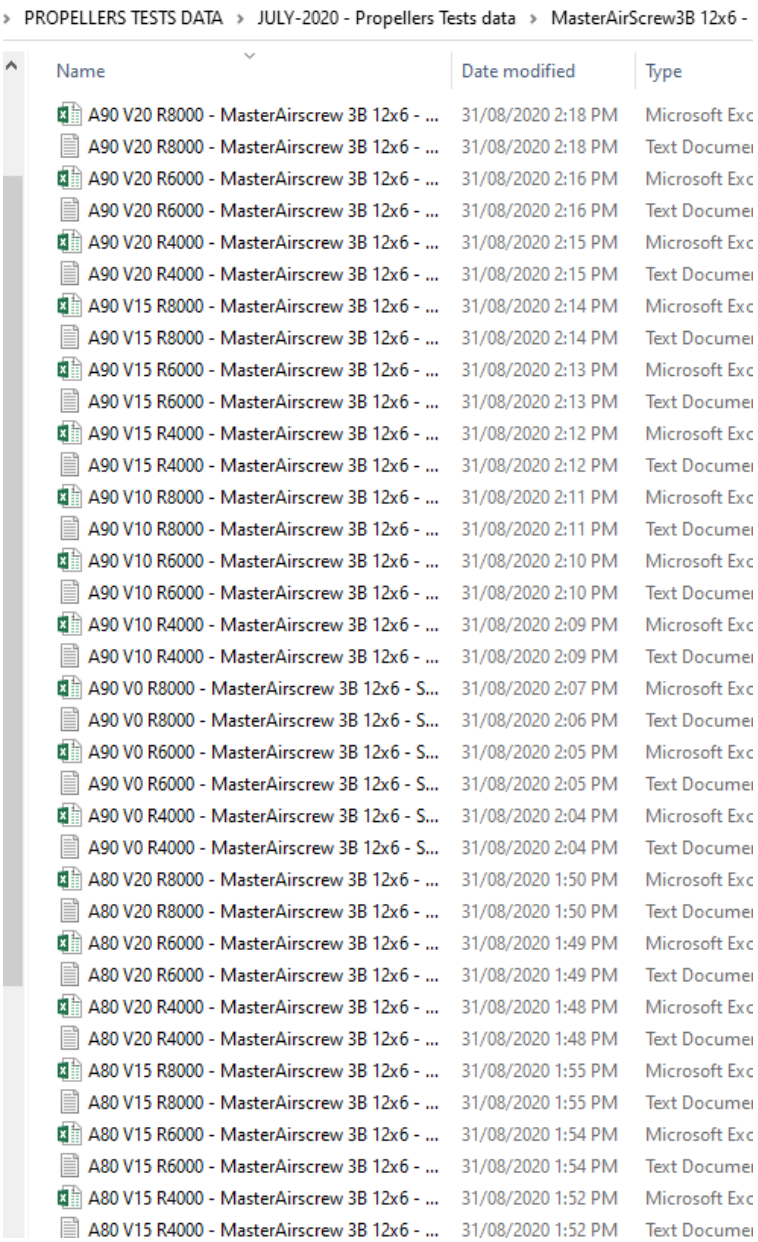


Figure B.3: Screenshot of the folder containing .txt and .xlxs data files from propellers tests

B.3 Script to analyse the data compiled and the class "propeller.m"

The script "PROPELLER_scriptToAnalyseDATA.mlx", (see Appendix C.3), loads the files "CalibRIG- ..... _FITS.mat" and "*name_of_propeller*_ResultAcquired.mat", with its table. The propeller gross forces measured that are stored in the "ResultAcquired_Table", $GF_x, GF_y, GF_z, GM_x, GM_y, GM_z$, are netted from the calibration rig forces using the loaded fits. Net thrust, normal force and moments are then estimated for every condition according to the geometry of the holding rig. A new table named "Table_PreparedTestsData", with $AoA, V, T, N_p, M_p, Q_p, M_{yaw}, E_{pow}$ data is created to be input, as a variable, into a MATLAB class object created and named "propeller.m", (see Appendix C.4).

All the calculations relevant to the propeller analysis, as well as graphs are produced by the class object, that bears properties designed according to momentum theory and the new theory. For instance, thrust values acquired for every test condition, that are passed as class input properties alongside the other variables in the table "Table_PreparedTestsData", allows for the calculation of induced speeds w according to Eq. (2.21). Once w is obtained, then all the remaining calculations can be performed. The class also features various public methods than can be called for calculations not restrained to the object created. The output is returned to the script for further analysis.

One of the propeller class public functions "FUNcreateClusters" identifies and organizes data according to a desired number of clusters for RPM, V, J values. This is done as for different propellers, experiments will be aimed at different approximate values of RPM, and V conditions, and so the code can be able to identify these target values automatically. For example, if tests were based on 4 RPM values, there should be 4 clusters for RPM chosen as number of clusters input. The same procedure is done for V. For J, which is a dependable variable and not previously set, a number of 9 clusters was elected for the code to choose, around the most frequent values. The function output are indices that indicates to which cluster every result line belongs.

A main table with all variables of interest from the results is created and named "Table_ResultsAnalysed". This table contains 56 columns of variables and includes

the vectors of cluster indices to RPM, V, J.

The function "FUNcreateSubTables", from class propeller, then creates many sub-tables for constant AoA, RPM, V, and J values of interest, using the cluster indices relative to each average value of every cluster. For instance, if tests were performed at around 4 RPM values, then 4 sub-tables with data relative to each RPM value are created, and so on, for V, J, and AoA. Table B.1 shows an example of the script output for a propeller tested for nominal RPM values of 3,000, 4,000, 5,000 and 6,000. The actual tested data collected around those values, captured by the cluster indices, are shown below tablesRPM in the table. The AoA values are shown under tablesAoA. As the AoA values are fixed and not measured, there is no need to average them. Tables for V and J are also depicted. All tables are stored in cell arrays, that contain the variable average value as reference for each cluster, in the second column.

Examples of sub-tables contained in the cell arrays are illustrated in Table B.2, for AoA, and V, and in Table B.3, for RPM, and J, where some of the 56 columns of calculated variables are shown. The sub-tables and the main table "Table_ResultsAnalysed', obtained in the calculations, will be used to create many plots and fits by the script. All fits, plots and tables are saved in a file named "*name_of_propeller_*-ResultsAnalysed.mat".

tablesAoA = 7×2 cell array	
{14×56 table}	{[0]}
{13×56 table}	{[30]}
{ 8×56 table}	{[45]}
{10×56 table}	{[55]}
{11×56 table}	{[65]}
{15×56 table}	{[75]}
{15×56 table}	{[90]}
tablesRPM = 4×2 cell array	
{24×56 table}	{[3.2832e+03]}
{27×56 table}	{[5.2475e+03]}
{16×56 table}	{[7.0890e+03]}
{19×56 table}	{[8.2898e+03]}
tablesV = 4×2 cell array	
{ 3×56 table}	{[0]}
{29×56 table}	{[9.8172]}
{25×56 table}	{[13.2700]}
{29×56 table}	{[16.8439]}
tablesJ = 8×2 cell array	
{ 3×56 table}	{[0]}
{ 6×56 table}	{[0.1931]}
{17×56 table}	{[0.2840]}
{14×56 table}	{[0.3183]}
{19×56 table}	{[0.3856]}
{14×56 table}	{[0.4917]}
{ 8×56 table}	{[0.6848]}
{ 5×56 table}	{[0.8686]}

Table B.1: Cell arrays containing sub-tables created by the script "PROPELLER_scriptToAnalyseDATA.m"

Table B.2: Examples of cell arrays containing sub-tables of AoA and V. Sub-tables for $AoA = 60^o$, and $V = 15$ are shown, with some of the 56 columns of calculated variables.

Table B.3: Examples of cell arrays containing sub-tables of RPM and J. Sub-tables for $RPM = 6,000$, and $J = 0.50$ are shown, with some of the 56 columns of calculated variables.

Appendix C: The Matlab codes

C.1 The script to compile and analyse rig forces

```
% Script " Read_CalibrationRIG_TestFiles_Script.m"
% script to open all files in a folder, compile data and perform average
% and stdev calculations, to find fits for forces and plot graphs

% path='P:\RAFAEL\WIND TUNNEL & MATLAB\PROPELLERS TESTS DATA\
% JULY-2020 - Propellers Tests data\Calibration New holder AXI2835-10mm keysteel';
% path = 'P:\RAFAEL\WIND TUNNEL & MATLAB\PROPELLERS TESTS DATA\
% JULY-2020 - Propellers Tests data\Calibration New holder Cobra2826-6mm keysteel';
path1 = 'P:\RAFAEL\WIND TUNNEL & MATLAB\PROPELLERS TESTS DATA';
path2 = 'JULY-2020 - Propellers Tests data\';
path3 = 'Calibration - Sunnysky Motor Kv920 - 10mm keysteel';

d=dir(fullfile(path1,path2,path3,'*.txt'));
                    % dir lists the files in the current working directory
file_names={d.name};  % file_names is string
file_names = file_names'; % changing from default line vector to column vector

n = numel(file_names); % numel = number of elements i.e. number of files in folder
cellArray = cell(n,4);
    % creates a cell array which mixes different types, doubles, string..;
angle = cell(n,1);   % allocates a predefined space for cell created from
                        % part of file name which is the AoA (string format)

for k=1:n  %numel = number of elements i.e. number of files in folder
 f=fullfile(path,file_names{k});
 fopen('file_names','r');    % fid identifies status and fopen opens files
                             % fopen is for txt files

 dataForces = table2array( readtable(f));  % reading the file which is a table
                                        % cause format.txt and converting to array
     dataForces(:,1)=[];  % deleting first colums of time stamp
     meanValues_Forces = mean(dataForces,'omitnan');
     stdValues_Forces = sqrt(var(dataForces)) ;
   %  stdValuesPCT_Forces = stdValues_Forces./abs(meanValues_Forces)*100;

 location_of_A = strfind(file_names{k},'A'); % finds where first letter A is
                     % located in filename (which in this case is 1st place)
                     % cannot use a file name with A prior to A used to identify
                     % angle ex.. A30
 angle{k} = (file_names{k}(location_of_A+1:location_of_A+2)); % this cell will
               % contain the value of AoA read from file name in string format

 cellArray (k,:,:,:)= {file_names{k}, angle{k}, meanValues_Forces, ...
                                                        stdValues_Forces };
                         % creating a cell array which contains various types

end

angle = str2double(angle);nban
res = cell2mat([cellArray(:,3)  cellArray(:,4)]);
```

```
AA = [angle res];
AA(isnan(AA))=0;  % SUBSTITUES NaN VALUES FOR ZERO. MOSTLY V->0 GIVES THIS ERROR


format bank  %short  %shortg

CALIBRATIONRig_Table = [cell2table(file_names)  array2table(AA, 'VariableNames',...
    {'AoA','V','RigFx','RigFy','RigFz','RigMx','RigMy','RigMz','stdV',...
    'stdRigFx','stdRigFy','stdRigFz','stdRigMx','stdRigMy','stdRigMz' })];

AoA =    CALIBRATIONRig_Table.AoA;
V =      CALIBRATIONRig_Table.V;
RigFx = CALIBRATIONRig_Table.RigFx;
RigFy = -CALIBRATIONRig_Table.RigFy; % wrong orientation on Windtunnel Labview code
RigFz = CALIBRATIONRig_Table.RigFz;
RigMx = CALIBRATIONRig_Table.RigMx;
RigMy = -CALIBRATIONRig_Table.RigMy; % wrong orientation on Windtunnel Labview code
RigMz = CALIBRATIONRig_Table.RigMz;

%% Initialization.

% Initialize arrays to store fits and goodness-of-fit.
fitresult = cell( 6, 1 );
gof = struct( 'sse', cell( 6, 1 ), ...
    'rsquare', [], 'dfe', [], 'adjrsquare', [], 'rmse', [] );

%% Fit: 'RigFx'.
[xData, yData, zData] = prepareSurfaceData( AoA, V, RigFx );

% Set up fittype and options.
ft = 'thinplateinterp';

% Fit model to data.
[fitresult{1}, gof(1)] = fit( [xData, yData], zData, ft, 'Normalize', 'on' );

% Plot fit with data.
GH(1) = figure( 'Name', 'RigFx' );
h = plot( fitresult{1}, [xData, yData], zData );
legend( h, 'RigFx', 'RigFx vs. AoA, V', 'Location', 'NorthEast' );
% Label axes
xlabel AoA
ylabel V
zlabel RigFx
grid on

%% Fit: 'RigFy'.
[xData, yData, zData] = prepareSurfaceData( AoA, V, RigFy );

% Set up fittype and options.
ft = 'thinplateinterp';

% Fit model to data.
[fitresult{2}, gof(2)] = fit( [xData, yData], zData, ft, 'Normalize', 'on' );

% Plot fit with data.
GH(2) = figure( 'Name', 'RigFy' );
h = plot( fitresult{2}, [xData, yData], zData );
legend( h, 'RigFy', 'RigFy vs. AoA, V', 'Location', 'NorthEast' );
% Label axes
xlabel AoA
ylabel V
zlabel RigFy
grid on

%% Fit: 'RigFz'.
[xData, yData, zData] = prepareSurfaceData( AoA, V, RigFz );
```

```
% Set up fittype and options.
ft = 'thinplateinterp';

% Fit model to data.
[fitresult{3}, gof(3)] = fit( [xData, yData], zData, ft, 'Normalize', 'on' );

% Plot fit with data.
GH(3) = figure( 'Name', 'RigFz' );
h = plot( fitresult{3}, [xData, yData], zData );
legend( h, 'RigFz', 'RigFz vs. AoA, V', 'Location', 'NorthEast' );
% Label axes
xlabel AoA
ylabel V
zlabel RigFz
grid on

%% Fit: 'RigMx'.
[xData, yData, zData] = prepareSurfaceData( AoA, V, RigMx );

% Set up fittype and options.
ft = 'thinplateinterp';

% Fit model to data.
[fitresult{4}, gof(4)] = fit( [xData, yData], zData, ft, 'Normalize', 'on' );

% Plot fit with data.
GH(4) = figure( 'Name', 'RigMx' );
h = plot( fitresult{4}, [xData, yData], zData );
legend( h, 'RigMx', 'RigMx vs. AoA, V', 'Location', 'NorthEast' );
% Label axes
xlabel AoA
ylabel V
zlabel RigMx
grid on

%% Fit: 'RigMy'.
[xData, yData, zData] = prepareSurfaceData( AoA, V, RigMy );

% Set up fittype and options.
ft = 'thinplateinterp';

% Fit model to data.
[fitresult{5}, gof(5)] = fit( [xData, yData], zData, ft, 'Normalize', 'on' );

% Plot fit with data.
GH(5) = figure( 'Name', 'RigMy' );
h = plot( fitresult{5}, [xData, yData], zData );
legend( h, 'RigMy', 'RigMy vs. AoA, V', 'Location', 'NorthEast' );
% Label axes
xlabel AoA
ylabel V
zlabel RigMy
grid on

%% Fit: 'RigMz'.
[xData, yData, zData] = prepareSurfaceData( AoA, V, RigMz );

% Set up fittype and options.
ft = 'thinplateinterp';

% Fit model to data.
[fitresult{6}, gof(6)] = fit( [xData, yData], zData, ft, 'Normalize', 'on' );

% Plot fit with data.
GH(6)= figure( 'Name', 'RigMz' );
h = plot( fitresult{6}, [xData, yData], zData );
legend( h, 'RigMz', 'RigMz vs. AoA, V', 'Location', 'NorthEast' );
```

```
% Label axes
xlabel AoA
ylabel V
zlabel RigMz
grid on

%% defining other names for the fits and saving to file

RigFx_fit = fitresult{1};
RigFy_fit = fitresult{2};
RigFz_fit = fitresult{3};
RigMx_fit = fitresult{4};
RigMy_fit = fitresult{5};
RigMz_fit = fitresult{6};

%saving figures to file
path1 = 'P:\RAFAEL\WIND TUNNEL & MATLAB\MATLAB ANALYSIS - Wind tunnel';
path2 = 'Functions\FIT DATA FUNCTIONS';
filenm = 'CalibRIG-SunnyskyMotorKV920-10mm keysteel_FITS.mat';
filenameFITS = fullfile(path1,path2,filenm);

save(filenameFITS,'RigFx_fit','RigFy_fit','RigFz_fit','RigMx_fit','RigMy_fit',...
                                                        'RigMz_fit')
path3 =  'Plots and CurveFittings\fig_MATLAB\PROP TESTS JULY20';
filenamefig = fullfile(path1,path3,'CalibRIG_SunnyskyMotorKV920_graphs.fig');
savefig(GH,filenamefig);

savedpath1 = 'P:\RAFAEL\WIND TUNNEL & MATLAB\MATLAB ANALYSIS - Wind tunnel';
savedpath2 = 'Tables and Arrays saved\ResultsAcquired';
                                %get the directory of input files
savedfilename = 'CALIBRATION_RIG_SunnyskyMotorKV920_ResultAcquired';
save (fullfile(savedpath1,savedpath2, savedfilename),'CALIBRATIONRig_Table')

%clearvars -except CALIBRATIONRig_AXI2835_Motor_Table
%fclose('all');
```

C.2 The script to read and compile tests data

```
% Script "Read_Prop_Test_DATA.m"
% script to open all files in a folder, FORCES FILEs (.txt) and POWER FILES (.xlxs)
% to compile data and perform average and stdev calculations

% path=('P:\LATEST IMPORTED OCT19\RAFAEL\WIND TUNNEL\JULY2020\
%                                          APC18x5.5 - Motor AXI2835-10');
% path=('P:\RAFAEL\WIND TUNNEL & MATLAB\PROPELLERS TESTS DATA\
%           JULY-2020 - Propellers Tests data\Gemfan10x10 - Motor Sunnysky KV920');
corepath = 'P:\RAFAEL\WIND TUNNEL & MATLAB\PROPELLERS TESTS DATA';
path1 = 'JULY-2020 - Propellers Tests data';
folder = 'MasterAirScrew3B 12x6 - SunnySkyMotor 2820 KV920';
list_txt = dir(fullfile(corepath, path1, folder,'*.txt'));
           %dir lists the .txt files in the current working directory
filenames_txt={list_txt.name};  %file_names is string
filenames_txt = filenames_txt';%changing from default line vector to column vector

list_xlsx = dir(fullfile( corepath, path1, folder,'*.xlsx'));
           % dir lists the .xlsx files in the current working directory
filenames_xlsx ={list_xlsx.name};  %file_names is string
filenames_xlsx = filenames_xlsx';
                        %changing from default line vector to column vector
n_txt = numel(filenames_txt); % numel = number of elements,i.e. of files in folder
n_xls = numel(filenames_xlsx);% numel = number of elements,i.e. of files in folder

if n_txt == n_xls
    n = n_txt;
```

```
else
    disp('NUMBER of txt files different than number of excel files in folder')
    return
end
fopen('filenames_txt','r');
fopen('filenames_xlsx','r');

cellArray = cell(n,6);
    % creates a cell array which mixes different types, doubles, string..;
angle = cell(n,1);
    % allocates a predefined space for cell created from part of file name
    % which is the AoA (string format)

for k=1:n
     k
 f_txt=fullfile(path,filenames_txt{k})
                            % chosing the kth file .txt and storing in f_txt
 [path, corename, ext] = fileparts(f_txt);
         % identifying its corename to search for the same corename file in xlsx
 dummy = dir(fullfile(path,strcat('*',corename,'*','xlsx')));
         % finds files in xlsx with 'corename' as part of the file name and
         % stores them in dummy
 f_xlsx = fullfile(path,dummy.name)
               %storing the equivalent corename in xlsx in f_xlsx

 dataForces = table2array( readtable(f_txt));   % reads data in txt file
                                                % and converts to double
 dataPower  = xlsread(f_xlsx);   % reads data in xlsx file

     dataForces(:,1)=[];  % deleting first colums of time stamp
     meanValues_Forces = mean(dataForces,'omitnan');
     stdValues_Forces = sqrt(var(dataForces,'omitnan')) ;
     meanValues_Power = mean(dataPower,'omitnan');
     stdValues_Power = sqrt(var(dataPower)) ;

 location_of_A = strfind(filenames_txt{k},'A');   % finds where letter A is
                   % located in filename (which in this case is 17th place)
 angle{k} = (filenames_txt{k}(location_of_A+1:location_of_A+2)); % this cell
          % will contain the value of AoA read from file name in string format

 %creating a cell array which contains various types
 cellArray (k,:,:,:,:)= {filenames_txt{k}, angle{k}, meanValues_Forces,...
                   stdValues_Forces, meanValues_Power, stdValues_Power };

end

angle = str2double(angle);
res = cell2mat([cellArray(:,3)  cellArray(:,4) cellArray(:,5) cellArray(:,6)]);
AA = [angle res];
AA(isnan(AA))=0;  % SUBSTITUES NaN VALUES FOR ZERO. MOSTLY AT V~0 WINDTUNNEL
                                                % SOFTWARE GIVES THIS ERROR

format bank  %short  %shortg

MasterAirScrew3B_12x6_ResultAcquired_Table = [cell2table(filenames_txt)...
    array2table(AA, 'VariableNames',{'AoA','V','GrossFx','GrossFy','GrossFz',...
    'GrossMx','GrossMy','GrossMz','stdV','stdGFx','stdGFy','stdGFz','stdGMx',...
    'stdGMy','stdGMz','Current','Voltage','Power_W','RPM','stdCurrent',...
    'stdVolt','stdPower','stdRPM' })];

savedpath1 = 'P:\RAFAEL\WIND TUNNEL & MATLAB\MATLAB ANALYSIS - Wind tunnel';
savedpath2 = 'Tables and Arrays saved\ResultsAcquired';
savedpath = fullfile(savedpath1,savedpath2);
savedfilename = 'MasterAirScrew3B_12x6_ResultAcquired';
save (fullfile(savedpath,savedfilename),'MasterAirScrew3B_12x6_ResultAcquired_Table')

% clear AA ans res k cellArray dataForces fileID meanValues_Forces ...
```

```
% stdValues_Forces stdValuesPCT_Forces dataPower;
% fclose('all');
```

C.3 The main script for the analysis

```
%  " PROPELLER_scriptToAnalyseDATA.m "
% Script to analyse the data acquired for Propellers TESTS
% NEW TEST RIG with turntable moving.
cd 'P:\RAFAEL\WIND TUNNEL & MATLAB\MATLAB ANALYSIS - Wind tunnel'
corepath = 'P:\RAFAEL\WIND TUNNEL & MATLAB\MATLAB ANALYSIS - Wind tunnel';
addpath(genpath(corepath))  % add path of any subdirectory below 'Wind tunnel'
% genpath command load('APC18x55_ResultAcquired.mat','APC18x55_ResultAcquired_Table');

load('CalibRig_CobraMotor2826_FITS')
% load('CalibRIG-SunnyskyMotorKV920-10mm keysteel_FITS.mat')
            % loading the mat file with curve FITS for RIGCalibration
            % both CALIB results practically the same, also tested with
            % propellers result

 load('APC18x55_ResultAcquired_WO_OUTLIERS','APC18x55_ResultAcquired_Table');
% load('APC15x4_ResultAcquired_WO_OUTLIERS.mat','APC15x4_ResultAcquired_Table');
% load('Prop10x10_ResultAcquired_WO_OUTLIERS.mat','Prop10x10_ResultAcquired_Table');
% load('MasterAirScrew3B_12x6_ResultAcquired_WO_OUTLIERS.mat',...
                    %         'MasterAirScrew3B_12x6_ResultAcquired_Table');

h1 = 0.44; %  [m]  % height of the column from the WT floor
b1 = 0.235; %  [m] % distance from force-balance centre to column
b2 = 0.320; %  [m] % distance from propeller to column
%#################################################################
Din = 18;    % DON'T FORGET TO INPUT PROPELLER DIAMETER in INCHES
Name = 'APC18x55';  % SET PROPELLER NAME  'MAS-3B12x6.0'
% Tab = MasterAirScrew3B_12x6_ResultAcquired_Table;  % DON'T FORGET TO CHANGE NAME
Tab = APC18x55_ResultAcquired_Table;
%#################################################################
  % replaces any NaN for zero in the table -mostly for V

% indx = Tab.RPM > 14000 & Tab.RPM < 16000;
% option to analyse choosing only one RPM
% Tab = Tab(indx,:);

n = height(Tab);

%Tab(:,1)=[]; % disregardingfiles names (still recorded on Resultacquired file)
A = Tab.AoA; % results read from data acquired at windtunnel
V = Tab.V;
GFx =  Tab.GrossFx;
GFy = -Tab.GrossFy; % WT program w/ wrong orientation definition of y axis
GFz =  Tab.GrossFz;
GMx =  Tab.GrossMx;
GMy = -Tab.GrossMy; % WT program w/ wrong orientation definition of y axis
GMz =  Tab.GrossMz;
stdV =   Tab.stdV;
stdGFx = Tab.stdGFx;
stdGFy = Tab.stdGFy;
stdGFz = Tab.stdGFz;
stdGMx = Tab.stdGMx;
stdGMy = Tab.stdGMy;
stdGMz = Tab.stdGMz;
amp =    Tab.Current;
Volt =   Tab.Voltage;
EPow =   Tab.Power_W;
RPM =    Tab.RPM;
stdAmp = Tab.stdCurrent;
stdVolt = Tab.stdVolt;
```

```
stdPow = Tab.stdPower;
stdRPM = Tab.stdRPM;
%-------------------------------------------------------
% net forces measured by Force-balance
NetFx = GFx - RigFx_fit(A,V);% Rig fit is fitcurve acquired for calibration
NetFy = GFy - RigFy_fit(A,V);% RigFy_fit already corrected in the fitcurve x(-1)
NetFz = GFz - RigFz_fit(A,V);
NetMx = GMx - RigMx_fit(A,V);
NetMy = GMy - RigMy_fit(A,V);% RigMy_fit already corrected in the fitcurve x(-1)
NetMz = GMz - RigMz_fit(A,V);

% propeller calculated forces FROM WIND TUNNEL RIG GEOMETRY
T =   NetFx;
Np =  NetFy;  % propeller Normal force - positive defined in yaxis direction
Qp  = NetMx + h1*NetFy;     % propeller torque - positive following x axis

Myaw = + NetMy -h1*NetFx  + (b2-b1) * NetFz; % propeller yaw moment (yaxis)
                             % cause:  NetMy = NetFx*h1-Netfz*(b2-b1)+Myaw
Mp = NetMz - (b2-b1) * NetFy; % propeller pitch - negative following z axis
% definition as dAoA increased with -z (could have turned AoA in opposite way)

% T = T.*(T>0); % multiply T by logical test (1 if T>0 and or if T<0)i.e.
                                          % zeroing negative values of T
V = V.*(V>6); % cleaning noise data for V=0 showing slow speeds measurements

PreparedTestsData = [A V RPM T Np Mp Qp Myaw EPow];
Table_PreparedTestsData = array2table(PreparedTestsData,'VariableNames',...
                {'A',' V',' RPM','T',' Np',' Mp',' Qp',' Myaw',' EPow'});

PropObj = propeller(Din,Name,Table_PreparedTestsData);
           % creating a variable PropObj of class propeller

stdT = stdGFx;
stdNp = stdGFy;
stdRPMpct = 100*stdRPM./RPM;
stdVpct = 100*stdV./V;
stdJpct = 100*propeller.errJ_pct(stdV./V, stdRPM./RPM);
stdCTpct = 100*propeller.errCT_pct(stdT./T, stdRPM./RPM);
stdWpct  = 100*propeller.errW_pct(stdT./T,stdV./V,PropObj);

E =         PropObj.E;
E_ult=      PropObj.E_ult;
e =         PropObj.e;  %entrainment factor = 1/cosE
D =         PropObj.D;
Rho =       PropObj.Rho;
Aslp =      PropObj.Aslp;
Aslp_ult =  PropObj.Aslp_ult;
w =         PropObj.w;
w_V=        w./V;
Taxial =    PropObj.Taxial;
Twing =     PropObj.Twing;
Vd =        PropObj.Vdisk;
Vd_ax =     PropObj.Vd_ax;
Vd_eN =     PropObj.Vd_eN;
Vult =      PropObj.Vult;
Ct =        PropObj.Ct;
CT =        PropObj.CT;
Cn =        PropObj.Cn;
CN =        PropObj.CN;
Cm =        PropObj.Cm;
CM =        PropObj.CM;
J =         PropObj.J;
eta=        PropObj.eta;
etaMT=      PropObj.etaMT;
UsefulPow = PropObj.UsefulPow; % output power = T*V*cos(A)
ReqPow =    PropObj.ReqPow; % input power = Required power = Q*Omega
TotalIdealPow =  PropObj.TotalIdealPow; % input power = Ideal power from MT
```

```
WF =          PropObj.WF;

etaEPow = UsefulPow./EPow; % for comparison, considering
                                                % input power = electric power
Tw_Ta =    Twing./Taxial;
%-------------------------------------------------------------------------
Vd_eq = 0.5*cosd(E).^-1.*(Vult.^2-V.^2)./(Vult.*cosd(E_ult)-V.*cosd(A));
% Vdisk from equation derived from power equation. thesis

%-------------------------------------------------------------------------
%calculation for Model that projects T(AoA,RPM,V) based on Thrust data
%measured at AoA=0

[T_Ao, GofT_Ao] = propeller.fitT_Ao(PropObj);
   % fitT_Ao returns the function fit and gof for T=f(RPM,V) at AOA=0.
   % arguments of T_Ao are RPM,V

w_VprojT = propeller.w_V_projT (Din, A, V, T_Ao(RPM,V));
Tmodel_proj = propeller.T_projected(Din, T_Ao(RPM,V), V, A);

[CT_Ao, GofCT_Ao]= propeller.fitCT_Ao(PropObj);  % CT_Ao is the function fit
                                                 % for CT at AOA=0. -argument is J
w_VprojCT = propeller.w_V_projCT (CT_Ao(J), J, A);
CTmodel_proj = propeller.CT_projected (CT_Ao(J), J, A);

format bank
%-------------------------------------------------------------------------
% partition data of V, RPM and J into clusters to create subtables further on

[idxRPM,Cent_RPM,idxV,Cent_V,idxJ,Cent_J] = ...
                                    propeller.FUNcreateClusters(RPM,4,V,4,J,9);

%-------------------------------------------------------------------------
ResultsAnalysed = ...
   [A RPM  V J idxRPM  idxV  idxJ Tw_Ta Twing Taxial T  Tmodel_proj w  w_V ...
        w_VprojT w_VprojCT CT CTmodel_proj  Ct Np  Cn CN Vd_eq Vd ...
        Vd_ax Vd_eN E_ult E e Aslp Aslp_ult Vult Cm CM Mp Myaw Qp etaMT eta etaEPow...
        TotalIdealPow ReqPow EPow  Volt amp stdV stdVpct...
        stdGFx stdGFz stdT stdNp stdRPMpct stdJpct stdCTpct stdWpct stdPow];

Table_ResultsAnalysed = array2table(ResultsAnalysed,'VariableNames',...
    {'AoA','RPM','V','J','idxRPM','idxV','idxJ','Twing_Taxial','Twing',...
    'Taxial','T','Tmodel_projected','w','w_V','w_VprojT','w_VprojCT','CT',...
    'CTmodel_projected','Ct','Np','Cn','CN','Vd_eq','Vd',...
    'Vd_ax','Vd_eN','E_ult','E','e','Aslp','Asp_ult','Vult','Cm','CM',...
    'Mp','Myaw','Qp','etaMT','eta','etaEPow','TotalIdealPow','ReqPow',...
    'EPow','Volt','Amp','stdV','stdVpct','stdGFx',' stdGFz','stdT',...
    'stdNp','stdRPMpct','stdJpct','stdCTpct','stdWpct','stdPow'});
%-------------------------------------------------------------------------
%  creating subtables for all AoA, RPM, V, J

[tablesAoA, tablesRPM, tablesV, tablesJ] = ...
                        propeller.FUNcreateSubTables(Table_ResultsAnalysed)

%-------------------------------------------------------------------------
%  calling fit and plot functions

figT_AoA0  = propeller.plotT_AoAo(PropObj);
figCT_AoA0 = propeller.plotCT_AoAo(PropObj);

[fitsCTxJ, gofsCTxJ, figCTxJ_As] = ...
                        propeller.Plot_CTxJ_variousAoA(PropObj,tablesAoA);

[fits_wV_x_J, gofs_wV_x_J] = propeller.fits_wV_from_J_variousAs(tablesAoA);

[ InterpSurf_J_from_AoA_wV, ...
            InterpSurf_wV_from_AoA_J, figs_wVxJ_As]...
```

```
                    = propeller.Plot_wV_vs_J_variousAoA(PropObj,tablesAoA);

figTwTa_x_wV = propeller.Plot_TwTax_vs_wV_variousAoA(PropObj, tablesAoA);

[ fig_TwTa_x_J_As ] =...
                propeller.Plot_TwTax_vs_J_variousAoA(PropObj, tablesAoA);
%-------------------------------------------------------------------------
% Surface   tested: 18x55 poly42,  10x10 poly33, MAS_3B12x6 poly44, 15x4 poly32
[fitCTxAxJ_Surf, gofCTxAxJ, fig_Surf_CTxAxJ,  fig_Mesh_CTxAxJ ] = ...
    propeller.Plot_Surf_CTxAxJ  (PropObj,InterpSurf_wV_from_AoA_J,'poly44');

    optionalHorzVectorAoAs = [0  30  45  60   70  76  80  90];
                                                % choosing angles for plot
[figsCT_CTa_CTw_xJ, figCTsxJ, figCToverJxJ] = ...
       propeller.Plot_CTs_x_J_variosA_fromfit  (PropObj, fitCTxAxJ_Surf,...
                        InterpSurf_wV_from_AoA_J,optionalHorzVectorAoAs);

    optionalHorzVectorJs = [ 0.2  0.3  0.4  0.5  0.6  0.8  1.0]; % (0.1:0.15:1.0);
[figCTsxAoA, figsCT_CTa_CTw_xAoA] = ...
      propeller.Plot_CTs_x_AoA_variosJ_fromfit  (PropObj,fitCTxAxJ_Surf  ,...
                          InterpSurf_wV_from_AoA_J,optionalHorzVectorJs);

     optionalHorzVectorJs = [ 0.1  0.2  0.3  0.4  0.5  0.6 ]; % choosing J values for plots
     ModelAngleShift = +15;        % choose shift anlge for simplified model
 [figsCT_withModelproj] =  propeller.Plot_CTs_x_AoA_vs_SimplifiedMODEL...
                       (PropObj, fitCTxAxJ_Surf ,InterpSurf_wV_from_AoA_J,...
                                    ModelAngleShift,optionalHorzVectorJs);

 [Errorfigs] =  propeller.ErrorPlots (PropObj,Table_ResultsAnalysed);

%-------------------------------------------------------------------------
%  saving results

Resultpath = fullfile(corepath,'\Tables and Arrays saved\ResultsAnalysed\');
FolderName = Name;
mkdir(Resultpath,FolderName)  %creates a folder with propeller Name
                              % in parent folder Resultpath

savedfilename = strcat(Name,'_ResultsAnalysed.mat');
                                    % naming file according to prop tested
%   save (fullfile(Resultpath,FolderName,savedfilename),'Name',...
%     'Table_PreparedTestsData','Table_ResultsAnalysed','tablesAoA',...
%      'tablesRPM','tablesV', 'tablesJ','fitsCTxJ','fits_wV_x_J',...
%     'gofs_wV_x_J','fitCTxAxJ_Surf','gofCTxAxJ','InterpSurf_J_from_AoA_wV', ...
%       'InterpSurf_wV_from_AoA_J');

Allfigs = [figT_AoA0, figCT_AoA0, figsCT_CTa_CTw_xJ, figCTsxJ, ...
           figCToverJxJ, figCTsxAoA, figsCT_CTa_CTw_xAoA,...
            figsCT_withModelproj, Errorfigs];

% savefig(Allfigs,fullfile(Resultpath,FolderName,[Name,'_plots.fig']));


% clear all variable from workspace except those which names start with
% Table,...etc
% clearvars -except Table* tables*  PropObj* fit* gof* Cent* InterpFunc*...
%         Name id* fig*
```

C.4 The class "propeller.m"

```
classdef propeller
   % CLASS PROPELLER

    properties  (Constant)
```

```
    Rho = 1.21  % density of air at sea level and 19 deg C.
                % can be changed externally
end

properties
    Din
    Name
    RPM
    V
    AoA     % propeller angle of incidence in deg
    T       % Thrust
    Np      % propeller force Normal to Thrust when at incidence
    Qp      % propeller + motor torque
    Mp      % propeller pitch moment (about pitch axis) - due to
                    % Thrust asymmetry at incidence (not 1-Np)
    Myaw    % propeller yaw moment (about yaw axis) - due to
              % Thrust asymmetry at incidence
    EPow    % electric power measured before ESC
    inputHandleThrustFunction  %enter the name of the function that
            %interpolates Thrust with @before the name - handle function

end
properties (Dependent)
            % dependent depends on other properties values.
                        % Calculated in getter function
            % protected does not show. private shows read only.
                        % both can't be changed
    Sdisk % swept area of propeller's disk
    J     % advance ratio  J = V / n.D
    Ct    % thrust coefficient from Momentum theory  Ct = T/q.Sdisk
    CT    % thrust coefficient as defined by  CT = T/Rho.n2.D4
    Cn    % Normal force coefficient, Momentum theory  Cn = Np/q.Sdisk
    CN    % Normal force coefficient as defined by  CN = Np/Rho.n2.D4
    Cq    % torque coefficient = Cq = Q/Rho.n2.D5
    Cm    % moment coefficient  Cm = My/(D*Sdisk*DynPress)
    CM    % moment coefficient  CM = My/(Rho*(RPM./60)^2*D^5)
    Cp    % power coefficient based on total or ideal power
                % Cp = T(VcosA+w)/Rho.n3.D5 and Cp = 2pi.Cq
    Vdisk % total Velocity at disk
    Vd_ax % Axial component of Vdisk
    Vd_eN % Normal component of Vdisk
    Vult  % ultimate slipstream speed
    w     % induced velocity at disk in axial direction
            % w_AoA0  % w solved from Matlab syms (analytical) solution
            % FOR AoA=0
            % case: w^4+2Vw^3+2w^2V^2 =(T/(2RhoSdisk))^2
            % largest of 4 roots: w = 1/2*sqrt(V^2 + 4T/(2Rho Sd))-V/2
            % w_V_AoA0  analytic w / V for AoA=0  from syms MATLAB
            % w/V = 1/2*sqrt[1+8/pi*CTo/J^2]-1/2       or
            % w/V = 1/2*sqrt[1+2T/(Rho*Sd*V^2)]-1/2 conferido os2 batem
    e           % entrainment factor  = Seff/Sdisk  = 1/cos(E)
    WF          % WingFactor  = sqrt((e-1)/(e+1))
                %             = sqrt((1-cosE)/(1+cosE))
    E           % epsilon = angle between Vdisk and T or w at disk
    E_ult       % ultimate angle between wult and Vult
                %    (wult is in axial direction as w and T)
    Aslp        % alpha slipstream at disk Aslp = AoA - E
    Aslp_ult    % ultimate alpha slipstream
    ReqPow      % ReqPow = Q.Omega = Q.2pi.n = Q.2pi.RPM/60
                % required or actual power is the power required to
                % spin the propeller
    UsefulPow   % useful power  T*V for axial flow  or T*Vcos(A) for
                % incidence McCormick V/Stol pg 126 or Aerod... pg345
    TotalIdealPow % Ideal power from momentum theory T*(VcosA+w)
                %      or T*Vd_eT   = T*Vd*cos(E)
                % ideal power is the maximum theoretical power
```

```
                    % achievable.
                    % power prediction and efficiency from momentum theory
                    % overestimated (no losses) McCormick V/Stol pg 126
                    % which would be eff = Usefulpower/TotalIdealPower
                    % in this case of MT analysis TotalIdealPow is the
                    % input power
        InducedPow  % T*w = induced power McCormick pg 345  (Aerodynamics,
                    %     Aeronautics and Flight mechanics)
        etaMT       % (ideal) efficiency = Output power/ input power
                    % from MT:  etaMT = Usefulpower / TotalIdealPow
                    %     etaMT = T.VcosA / T(VcosA+w)
                    %     as defined by Momentum theory
                    % in this case of MT analysis, input power considered
                    % is TotalIdealPow
        eta         % (Real) efficiency = Output power/ input power
                    % efficiency = Usefulpower / ReqPow (Qp*Omega)
                    %     eta = CT/(2pi*Cq)*J   = CT/Cp*J
        Dp_disk     % pressure jump at disk according to Momentum theory
        pd_minus    % this property returns the static pressure right
                    % before the disk from a symbolic function as
                    %     function of patm (neglecting change in Rho)
        pd_plus     % this property returns the static pressure right
                    %     after the disk
        Taxial      % 1st component of thrust (that disregards angle, ie.
                    %     epilon factor) w1st = w2nd = w
        Twing       % 2nd component of thrust (enhanced by epsilon factor)
        Nmodel      %FOR FUTURE INVESTIGATION. DOES NOT WORK AT HIGH ANGLES
    end

    properties (Hidden)
        D           % in m  (SI)
        DynPress    % dynamic pressure q = 0.5 Rho.V2
        sinA
        cosA
        wult
        n           % propeller frequency, revolutions per second  = RPM/60
    end

%--------------------------------------------------------------------------
    methods

        function obj = propeller(Din,Name, Table_PreparedTestsData)
            % constructor of the object. Constructs an instance of this
            % class not necessarily needs the input arguments
            % obj will be the name of the object defines of class propeller

           if nargin==3  % if number of arguments = 3 then assign the
                         % following,otherwise create object with no input
                         % properties immediately

                obj.Din          = Din;
                obj.Name         = Name;
                obj.AoA          = Table_PreparedTestsData.A;
                obj.V            = Table_PreparedTestsData.V;
                obj.RPM          = Table_PreparedTestsData.RPM;
                obj.T            = Table_PreparedTestsData.T;
                obj.Np           = Table_PreparedTestsData.Np;
                obj.Qp           = Table_PreparedTestsData.Qp;
                obj.Mp           = Table_PreparedTestsData.Mp;
                obj.Myaw         = Table_PreparedTestsData.Myaw;
                obj.EPow         = Table_PreparedTestsData.EPow;
            end
        end
        % getter functions define dependent properties
        function n = get.n(obj)
            n = (obj.RPM)/60;
        end
```

```
        function D = get.D(obj)  %D in [m]
            D = (obj.Din)*0.0254;
        end

        function cosA = get.cosA(obj)
            cosA = cosd(obj.AoA);
        end

        function sinA = get.sinA(obj)
            sinA = sind(obj.AoA);
        end

        function J = get.J(obj)
            J = (obj.V)./(obj.n*(obj.D));
        end

        function Sdisk = get.Sdisk(obj)
            Sdisk = pi*obj.D^2/4;
        end

        function DynPress = get.DynPress(obj)
            DynPress = 0.5*obj.Rho*(obj.V).^2;
        end

        function Ct = get.Ct(obj)
                % Propeller Thrust Coefficient based on dynamic pressure
%
                % momentum theory definition. Older definition still
                % used for windturbines as RPM is low
           Ct = obj.T./(obj.Sdisk*obj.DynPress) ;
        end

        function CT = get.CT(obj)
           CT = obj.T./(obj.Rho*(obj.n).^2*(obj.D).^4) ;
        end

        function Cn = get.Cn(obj)
           Cn =  (obj.Np)./(obj.Sdisk*obj.DynPress);
           %Propeller normal Force Coefficient based on dynamic pressure.
        end

        function CN = get.CN(obj) % Propeller normal Force Coefficient
            CN = (obj.Np)./ (obj.Rho*(obj.n).^2*(obj.D).^4);
        end

        function Cq = get.Cq(obj)
            Cq = (obj.Qp)./ (obj.Rho*(obj.n).^2*(obj.D).^5);
        end

        function Cm = get.Cm(obj)
            Cm = obj.Mp./(obj.D*obj.Sdisk*obj.DynPress);
        end

        function CM = get.CM(obj)
            CM = obj.Mp./(obj.Rho*(obj.n).^2*obj.D.^5);
        end

        function Cp = get.Cp(obj)
            Cp = obj.ReqPow./(obj.Rho*(obj.n).^3*obj.D.^5);
        end

        function TotalIdealPow = get.TotalIdealPow (obj)
          % Ideal or total power from momentum theory pg 214
          % Mccormick V/STOL - maximum theoretical power attainable
           TotalIdealPow = (obj.T).*((obj.V).*obj.cosA + obj.w);
        end
```

```
        function UsefulPow = get.UsefulPow (obj) % or propulsive power
            UsefulPow = obj.T.*obj.V.*obj.cosA;
        end

        function InducedPow = get.InducedPow (obj)
            InducedPow = obj.T.*obj.w;
        end

        function ReqPow = get.ReqPow (obj)
            ReqPow = abs(obj.Qp).*obj.RPM./60*2*pi;
        end

        function etaMT = get.etaMT(obj)
            etaMT = obj.UsefulPow./obj.TotalIdealPow;
            % tends to 1 as V->Inf
            % (https://web.mit.edu/16.unified/www/FALL/thermodynamics/
            % notes/node86.html#SECTION06374300000000000000
        end

        function eta = get.eta(obj)
            eta = obj.UsefulPow./obj.ReqPow;
        end
 %-------------------------------------------------------------------------
  function w = get.w(obj) % alternativelly (checked)
      %   SOLVES THROUGH POLYNOMIAL ROOTS FINDING (induced velocity)

      % FOR positive Thrust only.Not valid for T<0.If T<0 it returns zero
           leng = length (obj.T);
           w1_temp = zeros(leng,4);
           w = zeros(leng,1);
           c2 = 2*obj.V.*obj.cosA;

    for k = 1:1:leng
                if obj.T(k)<=0
                        w(k)=0;
                    continue
                end
    coefs = [1 c2(k) obj.V(k)^2  0  -(obj.T(k)/(2*obj.Rho*obj.Sdisk))^2 ];
    w1_temp(k,:,:,:)= real(roots(coefs)');  % roots calculate the roots
                  % of the 4th order equation(taking only real components)
           w(k) = max(w1_temp(k,:,:,:));
           % sometimes if all 4 roots are real, some will be negative,
           % or negative and zero. Taking the maximum guarantees
           % that will take the positive value
           w(k) = max(0,w(k)); % double guarantee that w >=0
     end
        %   this next line is wrong if T<0, cause w would be <0
        %   w = max(w(w>=0.0); %taking only positive values of w for T>0.
        %   This command can change the length of w, cause if all roots
        %   are negative it will decrease its length or if two roots are
        %   equal it will add more lines to the length. This command only
        %   works if there is always only one positive root.
  end
%-------------------------------------------------------------------------

        function Vd_ax = get.Vd_ax(obj)
            Vd_ax = obj.V.*obj.cosA + obj.w;
        end

        function Vd_eN = get.Vd_eN(obj)
            Vd_eN = obj.V.*obj.sinA;
        end

        function Vdisk = get.Vdisk(obj)
            Vdisk =((obj.Vd_ax).^2 + (obj.Vd_eN).^2 ).^0.5;
            %Vdisk from momentum theory - pg 213 McCormick
```

```
end

function E = get.E(obj)  %  output in [deg]
    % angle Epsilon, between T and Vdisk
    Erad = atan(obj.Vd_eN./obj.Vd_ax);
    E = rad2deg(Erad);
end

function WF = get.WF(obj)
    WF = sqrt((obj.e - 1)./(obj.e + 1));
end

function wult = get.wult(obj)
    wult = 2* obj.w;
end

function Vult = get.Vult(obj)
    Vult = sqrt (( obj.V + obj.wult.* obj.cosA).^2 + ...
                 ( obj.wult.* obj.sinA).^2 );
end

function Dp_disk = get.Dp_disk (obj)
    Dp_disk = 0.5 * obj.Rho * (obj.Vult.^2 - obj.V.^2);
end


function e = get.e(obj)  % entrainment factor
      e =  cosd(obj.E).^-1;
end

function Taxial = get.Taxial(obj)
    Taxial = 2*obj.Rho*obj.Sdisk*(obj.V.*obj.cosA + obj.w).*obj.w;
end

function Twing = get.Twing(obj)
    Twing = abs(obj.T) - abs(obj.Taxial);
end

function Nmodel = get.Nmodel(obj)  %Pi(Twing) = Twing*w = Np*VsinA
 %  DOES NOT WORK FOR HIGH ANGLES  model: Np = Twing*w /VsinA
    Nmodel = 2*obj.Rho*obj.Sdisk*(obj.w).^2.*obj.WF;
    Nmodel = sign(obj.Np).*Nmodel;
end

function E_ult = get.E_ult(obj)
    E_ult = propeller.calc_E(obj.wult,obj.V,obj.AoA);
    % where function calc_E returns value in degrees
end

function Aslp = get.Aslp (obj)
    Aslp = obj.AoA - obj.E;
end

function Aslp_ult = get.Aslp_ult(obj)
    w_overV = obj.w./obj.V;
    num = 2*w_overV.*obj.sinA;
    den = sqrt(1+4*w_overV.*obj.cosA + 4*(w_overV).^2);
    Aslp_ult = rad2deg(asin(num./den));
end


function pd_minus = get.pd_minus(obj)
    syms  f(patm)  %declaring f as a symbolic function of patm
    digits = 4; %number of displayed digits from vpa function
    f(patm) = patm + 0.5*obj.Rho*(obj.V.^2-obj.Vdisk.^2);  %vpa
     % variable-precision floating-point arithmetic (VPA)
     % to evaluate each element of the symbolic input x to at least
```

```
            % d significant digits, where d is the value of the digits
            % function
            pd_minus = vpa(f,digits); %in [Pa]
        end

        function pd_plus = get.pd_plus (obj)
            syms  f(patm)  %declaring f as a symbolic function of patm
            digits = 4; % number of displayed digits from vpa function
            f(patm) = obj.pd_minus + obj.Dp_disk;
            pd_plus = vpa(f,digits);

        end

    end
%-------------------------------------------------------------
    %##########################################################################
    %##########################################################################
    %##########################################################################

     methods (Access = public, Static)    %
           % Static methods are associated  with a class, but not with
           % specific instances of that class. These methods do not require
           % an object of the class as an input argument. Therefore,
           % static methods can be called without creating an object of the class.
           % to call a static method use nameofclass.method
           % instead of obj.method
      %-----------------------------------------------------
       function [y] = find_w (D_in,T,V,AoA)
           % checked - ALTERNATIVE WAY TO ROOT
           % finds solution for w of equation 18 AIAA paper
              if isnan(T)
                  %||abs(T)<=0.06
                  % for smaller values, fzero FUNCTION DOES NOT converge
                  % in some cases
                  y=0;
              else
                   if T<0
                      x0= -2.1;
                   else
                      x0= 2.5;
                   end
              fun = @(x) propeller.w_equation(x,D_in, T,V, AoA);
              x = fzero( fun, x0 );
              y = x;
              end
       end

       function y = w_equation (w,D_in, T,V, AoA)
              % equation 18 for T, from AIAA paper
              Rho = propeller.Rho;
              Sd = pi*(D_in*0.0254)^2/4;
              y = V.^2*w.^2 + 2*V.*cosd(AoA).*w.^3 + w.^4 - (T./2/Rho/Sd).^2;
                            % y needs to be zero for solving equation on w
       end
     %-----------------------------------------------------
       function [y] = find_w_V (CT,J,AoA)  %checked
                % finds solution for w/V of equation 19b AIAA paper
                   if J==0
                    y = Inf;
                   elseif isnan(CT)||CT==0
                    y=0;
                   else
                        if CT<0
                         x0= -5.1;
                        else
                         x0= 5.1;
                        end
```

```
                    fun = @(x) propeller.w_V_equation(x,CT,J,AoA);
                    x = fzero( fun, x0 );
                    y = x;
                   end
    end

    function y = w_V_equation (w_V,CT,J,AoA)
              % equation 19b for CT, from AIAA paper
         J2=J.^2;
         y = w_V.^2 + 2*cosd(AoA).*w_V.^3 + w_V.^4 - (2*CT./(pi/J2)).^2;
              % setting y to zero for solving equation on w_V
    end
  %-------------------------------------------------

    function [y_deg] = calc_E  (inducedspeed, V, AoAdeg)  % returns E
        % in degrees
              E_rad =  atan((V.*sind(AoAdeg))./(V.*cosd(AoAdeg) +...
                  inducedspeed));
              y_deg  = rad2deg(E_rad);
    end

    function y = WingFactor_fromEdeg(Edeg)   %tested e_factor = E_factor
              y = sqrt((1 - cosd(Edeg))./(1 + cosd(Edeg)));
    end

    function [y] = WingFactor (AoA, w_over_V) %as function of AoA, w/V.
        %checked with WF equation s funtion of e (checked)
        y = sind(AoA)./((sqrt(1 + 2*w_over_V.*cosd(AoA) + ...
            (w_over_V).^2)+cosd(AoA)+w_over_V));
    end

    function [y] = TwTax_from_wV (AoA, w_over_V)   % checked with
        % results Twing / Taxial
              y =  sind(AoA)./(cosd(AoA) + w_over_V).*...
                   propeller.WingFactor(AoA, w_over_V);
    end

    function [y] = TwTax_from_J (AoA, J, InterpFunc_wV_from_AoA_J)
        % BASED ON MESH EXTRAPOLATION FOR ANY ANGLE. PLOT NOT VERY GOOD.
        % checked with results Twing / Taxial
        if length(AoA)~=length(J)
          error ('length (AoA) must equal length(J)');
        end
              w_over_V = InterpFunc_wV_from_AoA_J(AoA,J);
              y =  sind(AoA)./(cosd(AoA) + w_over_V).*...
               propeller.WingFactor(AoA, w_over_V);
    end
%  ------------------------------------------------------------------------
    function [CT, CTax, CTwing] = CTax_CTwing_from_AoA_J ( AoA, J,...
                                  fitCTxA_J_Surf ,InterpFunc_wV_from_AoA_J)
                    %%%%%% BASIS FOR ALL ANALYSIS %%%%%%%
      % returns CT, CTax, CTwing calculated for a given AoA, J given the
      % propeler's fits of CT x AoA and J and the respective fit w/V x J
      % and surface data CT x J fit.

      if length(AoA)~=length(J)
          error ('length (AoA) must equal length(J)');
          % displays message and exits function
      end

      CT = fitCTxA_J_Surf(AoA,J);

      TwTa = propeller.TwTax_from_J (AoA, J, InterpFunc_wV_from_AoA_J);
      den = 1+TwTa;

      CTax = CT./den;
      CTwing = CT - CTax;
```

```
  end
%  ------------------------------------------------------------------------
  % CALCULATING w AND w/V at AoA=0  CHECKED with values w calculated
    function [y] = w_AoAo (Din,T,V)
        % solution of the quartic equation
        % found earlier via syms - syntatic solution
        % EQUATION 21 on the AIAA paper - NOT USED IN PROJECTION MODEL
        % w calculation from analytical solution from Matlab, FOR AoA=0
        % and T>=0, solution of Taxial =(T_AoA0) = 2*Rho*Sd[V(cosA=0)+w]w
        % RETURNS w, given T, V for AoA=0

     Sdisk = pi*(Din*0.0254)^2/4;
     y = max(0,real(0.5*sqrt(V.^2 + 4*(T./(2*propeller.Rho*Sdisk)))-V/2));
    end

    function [y] = w_V_AoAo (CT,J)
        % solution of the quartic equation found
        % earlier via syms - syntatic solution
        % EQUATION 22 on the AIAA paper - NOT USED IN PROJECTION MODEL
        % w calculation from analytical solution from Matlab % FOR AoA=0
        % and CT>=0, w/V calculation, does not work for static condition
        % V=0, J=0. RETURNS w/V, given CT, J for AoA=0
          if J==0
             y=Inf;
          else
             y = max(0,real(1/2*sqrt(8/pi*CT./J.^2 + 1)-0.5)); %checked
          end
    end
   %-------------------------------------------------------------
    function [f]=  w_V_projT (Din, T_AoAo, V, AoA) %checked for AoA=0
         % w/V FOR PROJECTION MODEL of Ttotal based on constant
         % Taxial = T_AoAo, one finds a w/V value at every AoA<>0,but
         % using Taxial= T_AoAo, which is the known measured  T at no
         % incidence. Assumption of the model is finding an aproximate T
         % as Taxial relatively constant over the AoA range.
         % EQ.31 paper AIAA

         % Solution of Taxial =(T_AoA0) = 2*Rho*Sd[VcosA + w]w    for
         % (w/V), ie (w/V) at AoA=0
         % BASKARA SOLUTION LEAVING IT AS FUNCTION OF COS(A) and not
         % using A=0 for projection.
         % EVERY T_AoAo IS ASSOCIATED WITH A VALUE OF V
             Sdisk = pi*(Din*0.0254)^2/4;
             Rho = propeller.Rho;
             T_AoAo = max(0,T_AoAo);
          f = max(0,real( 0.5*sqrt((cosd(AoA)).^2+4*T_AoAo./(...
                 2*Rho*Sdisk*V.^2))-0.5*cosd(AoA)));
    end

    function [f] = T_projected(Din, T_AoAo, V, AoA)
        % EVERY T_AoAo IS ASSOCIATED WITH A VALUE OF V
              % model for projecting T at a specific angle based on data
              % collected at AoA=0. I assume Taxial RELATIVELY constant
              % and equal to T_AoAo with the same RPM and V.
              % then I find w/V with that T_AoA=0,but with the formula of
              % Taxial using T(AoA=0) and leaving AoA as variable.
              % EQ31 on AIAA paper use this value of T and w/V as
              % function of cosA on the formula for
              % T = taxial (1+Twing/Taxial) eq. A1 on paper, T_Ao enters
              % extenally calling object fitT_AoA(RPM,V).
              % HOWEVER IN THE IN TABLE RESULTS IT IS NOT SMOOTH AS IT
              % CAPTURES DIFFERENT V AND RPMS (DIFFERENT T_AoA=0)AT AOA=0
              % FROM THE TABLE FOR EVERY POINT.
              % BETTER TO PROJECT BASED ON ANALYTIC FUNCTION
              T_AoAo = max(0,T_AoAo);
              w_V_proj = propeller.w_V_projT (Din, T_AoAo,V, AoA);
              f =    T_AoAo.*(1+sind(AoA)./(cosd(AoA)+w_V_proj)...
```

```
                .*propeller.WingFactor(AoA,w_V_proj));
    end
    %-------------------------------------------------------------

    function [f]=  w_V_projCT (CT_Ao, J, AoA) % checked for AoA=0
         % w/V FOR PROJECTION MODEL of CTtotal based on constant
              % CTaxial = CT_AoAo
              % one finds a w/V value at every AoA<>0, but using
              % CTaxial= CT_AoAo which is the known measured  CT at
              % no incidence.
              % Assumption of the model is finding an aproximate CT,
              % assuming CTaxial relatively constant over the AoA range.
              %  EQ. 31 paper AIAA
              % solution of CTaxial =(CT_AoA0) = 2*Rho*Sd[VcosA + w]w
              % for (w/V), ie (w/V) at AoA=0
              % BASKARA SOLUTION LEAVING IT AS FUNCTION OF COS(A) and not
              % using A=0 for projection
          CT_Ao = max(0,CT_Ao);
          f = max(0,real( 0.5*sqrt((cosd(AoA)).^2 + ...
                          (8/pi*J.^-2).*CT_Ao) -0.5*cosd(AoA)));
    end

    function [f] = CT_projected( CT_Ao, J, AoA )
          %model for projecting T at a specific angle based on data
              % collected at AoA=0. I assume Taxial RELATIVELY constant
              % and equal to T_AoAo with the same RPM and V.
              % then one finds w/V with that T_AoA=0,but with the formula
              % of Taxial using T(AoA=0) and leaving AoA as variable.
              % EQ31 on AIAA paper use this value of T and w/V as
              % function of cosA on the formula for
              % T = Taxial (1+Twing/Taxial) eq. A1 on paper.
              % T_Ao enters externally calling object fitT_AoA(RPM,V).
              % HOWEVER IN THE IN TABLE RESULTS IT IS NOT SMOOTH AS IT
              % CAPTURES DIFFERENT V AND RPMS (DIFFERENT T_AoA=0)
              % AT AOA=0 FROM THE TABLE FOR EVERY POINT.
              % BETTER TO PROJECT BASED ON ANALYTIC FUNCTION
                  CT_Ao = max(0,CT_Ao);
                  w_V_projCT = propeller.w_V_projCT ( CT_Ao,J, AoA);
              if J==0
                  f=0;
              else
                  f =   CT_Ao.*(1+sind(AoA)./(cosd(AoA)+w_V_projCT)...
                      .*propeller.WingFactor(AoA,w_V_projCT));
              end
     end
%-------------------------------------------------------------
%     ANALYTIC FUNCTIONS

    function [w_Vproj] = w_V_model_analytic (Din,T_AoAo, V)
                          % symbolic function of AoA for
                          % projecting w/V at any angle from V,
                          % T measured at A=0 (T_AoAo) and
                          % w/V at A=0 (w_V_AoAo)
          syms f(AoAdeg)
          digits = 3;
           Rho = propeller.Rho;
           Sdisk = pi*(Din*0.0254)^2/4;
          w_Vproj = max(0,real( 0.5*sqrt((cosd(AoAdeg)).^2+4*T_AoAo./(...
              2*Rho*Sdisk*V.^2))-0.5*cosd(AoAdeg)));
          w_Vproj=vpa(w_Vproj,digits);
    end

    function [ f ] = Tmodel_analytic(Din,T_AoAo, V)  %returns a
                          % symbolic function of AoA for
                          % projecting T at any angle from
                          % T measured at A=0 (T_AoAo) and V
```

```
                    % PLOTTED AND CHECKED
                    % ex: Tmodel = propeller.Tmodel_analytic(6,1.107,10)
                    % fplot(Tmodel,[0 90])
                    syms f(AoAdeg)
                    digits = 3;
            w_Vproj = propeller.w_V_model_analytic (Din,T_AoAo, V);

        f = T_AoAo.*(1+sind(AoAdeg)./(cosd(AoAdeg)+w_Vproj).*...
                    sind(AoAdeg)./(sqrt(1+2*cosd(AoAdeg)*w_Vproj+...
                    w_Vproj.^2) + cosd(AoAdeg)+ w_Vproj));
        f=vpa(f,digits);
    end
%-------------------------------------------------------------

   function [w_Vproj] = w_V_model_analyticCT (CT_AoAo, J, AngleShift)
                                % returns a SYMBOLIC FUNCTION of AoA for
                                % projecting w/V at any angle from V,
                                % CT measured at A=0 (CT_AoAo) and
                                % w/V at A=0 (w_V_AoAo)

   D = AngleShift; % shift in angle to try to approximate model to real wV
                     syms f(AoAdeg)
                     digits = 3;
      w_Vproj = max(0,real( 0.5*sqrt((cosd(AoAdeg+D)).^2+(8/pi/J.^2).*...
                                          CT_AoAo)-0.5*cosd(AoAdeg+D))));
      w_Vproj = vpa(w_Vproj,digits);
    end
 %-------------------------------------------------------------
    function [ f ] = CTmodel_analytic(CT_AoAo, J, AngleShift) % returns a
                      % SYMBOLIC FUNCTION of AoA for projecting T at
                      % any angle from T measured at A=0 (T_AoAo)and V
                   % ex: CTmodel = propeller.CTmodel_analytic(6,1.107,10)
                   %     fplot(CTmodel,[0 90])

    D = AngleShift;% shift in angle to try to approximate model to real CT
       syms f(AoAdeg)
       digits = 3;
       w_Vproj = propeller.w_V_model_analyticCT (CT_AoAo, J, AngleShift);

           f = CT_AoAo.*(1+sind(AoAdeg+D)./(cosd(AoAdeg+D)+w_Vproj).*...
                    sind(AoAdeg+D)./(sqrt(1+2*cosd(AoAdeg+D)*w_Vproj+...
                                w_Vproj.^2) + cosd(AoAdeg+D)+ w_Vproj));

           f=vpa(f,digits);
    end

%-------------------------------------------------------------

     function  [fitT_Ao_fromRPM_V, gof_fitT_Ao] = fitT_Ao (obj)
                                % returns a fit of T_AoA=0, vs RPM,V
              T_Ao = obj.T(obj.AoA==0);
              V_Ao = obj.V(obj.AoA==0);
              RPM_Ao = obj.RPM(obj.AoA==0);

              dummy = zeros(5,1);
        extraVs = [0 max(V_Ao)/4 max(V_Ao)/2 max(V_Ao)*0.75 max(V_Ao)]';

        RPM_Ao = [RPM_Ao;dummy] ;    % adding zero values of T, RPM
                                     % for the range of V_Ao
        V_Ao   = [V_Ao;extraVs] ;
        T_Ao   = [T_Ao;dummy] ;

       % opts = fitoptions( 'Method', 'ThinPlateInterpolant' );
       % [fitT_Ao, Gof_fitT_Ao] = fit([RPM_Ao,V_Ao],T_Ao,...
       %                              'thinplateinterp',opts);
       % opts.Normalize = 'on';
        opts = fitoptions('Normalize','on');
```

```
        [fitT_Ao_fromRPM_V, gof_fitT_Ao] = fit([RPM_Ao,V_Ao],T_Ao,...
                                                    'poly23',opts);
    end

    function [fitCT_Ao_fromJ, Gof_fitCT_Ao] = fitCT_Ao (obj)
                        %returns a fit of T_AoA=0, vs RPM,V
        CT_Ao = obj.CT(obj.AoA==0);
        J_Ao = obj.J(obj.AoA==0);
        CT_Ao = CT_Ao(J_Ao>0);
        J_Ao=J_Ao(J_Ao>0);
        opts = fitoptions('Normalize','on');
        [fitCT_Ao_fromJ, Gof_fitCT_Ao] = fit(J_Ao,CT_Ao,'poly2',opts);
    end
%-------------------------------------------------------------
%     ERROR ANALYSIS
    function [stdJ_pct] = errJ_pct (stdV_pct, stdRPM_pct)
        stdJ_pct    = sqrt((stdV_pct).^2+(stdRPM_pct).^2);
    end

    function [stdCT_pct] = errCT_pct (stdT_pct, stdRPM_pct)
        stdCT_pct    = sqrt((stdT_pct).^2 + 4*(stdRPM_pct).^2);
    end

    function [stdW_pct] = errW_pct (stdT_pct, stdV_pct,obj)
        V = obj.V;
        w=obj.w;
        A=obj.AoA;
        den = V.^2.*w + 3*V.*w.^2.*cosd(A) + 2*w.^3;
        num1 = V.^2.*w + 2*V.*w.^2.*cosd(A) + w.^3;
        num2 = V.^2.*w + V.*w.^2.*cosd(A);
        stdW_pct    = sqrt(((num1./den).*(stdT_pct)).^2 + ...
                           ((num2./den).*(stdV_pct)).^2);
    end

%-------------------------------------------------------------

 function [idxRPM,C_RPM,idxV,C_V,idxJ,C_J] = FUNcreateClusters...
     (VectorRPM,NumClustersRPM,VectorV,NumClustersV,VectorJ,NumClustersJ)
  % DATA PARTITION of V, RPM and J INTO CLUSTERS
  %
  % NumClusters is an input quantity of desired clusters. Based on data.
  % For example: if tests were based on 4 valued of RPM, there should be
  % 4 clusters. For J is more complicated as it is a variable that is an
  % output of other inputs. By looking at data it was chosen around 11
  % for the tests.
   format long
    opts = statset('Display','off');  % displays final iteration ['final'
                                      % instead of 'off'on display, ...
                                      % shows results of iterations]
    [idxRPM,C_RPM] = kmeans( VectorRPM, NumClustersRPM, 'Distance',...
      'cityblock','Replicates',10,'Options',opts);
        % ex: partition RPM into NumClustersRPM clusters ..10 iterations
        % idxRPM is the index of each cluster. C_RPM is the centroid
        % of each cluster
      opts = statset('Display','off');  % displays final iteration
      [idxV,C_V] = kmeans( VectorV, NumClustersV,'Distance',...
                          'cityblock','Replicates',8,'Options',opts);
                % partition V into NumClustersV  clusters ..6 iterations
      opts = statset('Display','off'); % displays final iteration
                                    % ['off' instead of 'final'no display]
      [idxJ,C_J] = kmeans( VectorJ, NumClustersJ,'Distance',...
                          'cityblock','Replicates',12,'Options',opts);
                % partition J into NumClustersJ clusters ..8 iterations
     format short
  end
    %-----------------------------------------------------------------------
```

```
function [tablesAoA, tablesRPM,...
        tablesV, tablesJ] = FUNcreateSubTables(Table_ResultsAnalysed)
    % This Function creates subtables for the clusters of AoA, RPM, V .
    % and J and stores them in arrays containing the tables and the
    % average value of every cluster

    %-------------------------------------------------------------
    % CREATING AoA TABLES

    ang = unique(Table_ResultsAnalysed.AoA);  % number of AoA instances
    k   = length(ang);                  % total number of AoA  instances
    tablesAoA = cell(k,2);      % creates cell array to store AoA tables

    for countA = 1: k
     TabAi = Table_ResultsAnalysed...
                           (Table_ResultsAnalysed.AoA==ang(countA),:);
                % creates a subtable for countA, i.e for a specific AoA
       tablesAoA{countA,1} = TabAi;    % stores TableAoAi of loop countA
       tablesAoA{countA,2} = ang(countA);
    end

    %-------------------------------------------------------------
    % CREATING RPM TABLES

    idxRPM = Table_ResultsAnalysed.idxRPM; % index of every RPM cluster
    MAXidxRPM = max(unique(idxRPM)); % total number of RPMclusters
    tablesRPM = cell(MAXidxRPM,2) ; % creates cell array to store RPM
                                                          % tables
    for countRPM = 1: MAXidxRPM
       TabRPMi = Table_ResultsAnalysed...
                           (Table_ResultsAnalysed.idxRPM==countRPM,:);
      % creates a subtable for countRPM, i.e for a specific RPM cluster
       Rot             = mean(TabRPMi.RPM); % average of an RPM cluster
       tablesRPM{countRPM,1} = TabRPMi; % stores TableRPMi of countRPM
                                                            % loop
       tablesRPM{countRPM,2} = Rot;
    end
      % arranging in RPM ascending order
    [~,idRPM] = sort([tablesRPM{:,2}], 'ascend');
    tablesRPM= tablesRPM(idRPM,:);

    %-------------------------------------------------------------
    % CREATING V TABLES

    idxV    = Table_ResultsAnalysed.idxV; %  index of every V cluster
    MAXidxV = max(unique(idxV));          % total number of Vclusters
    tablesV = cell(MAXidxV,2); % creates cell array to store V tables

    for countV = 1: MAXidxV

       TabVi = Table_ResultsAnalysed...
                              (Table_ResultsAnalysed.idxV==countV,:);
      % creates a subtable for countRPM, i.e for a specific RPM cluster
       Vmean = mean(TabVi.V);
       tablesV{countV,1} = TabVi; % stores TableVi of  countV loop
       tablesV{countV,2} = Vmean;
    end
    % arranging in ascending order of Vmean
    [~,idV] = sort([tablesV{:,2}], 'ascend');
    tablesV= tablesV(idV,:);

    %-------------------------------------------------------------
    % CREATING J TABLES

    idxJ    = Table_ResultsAnalysed.idxJ; % index of every J cluster
    MAXidxJ = max(unique(idxJ));          % total number of J clusters
    tablesJ = cell(MAXidxJ,2) ; % creates cell array to store J tables
```

```
        for countJ = 1: MAXidxJ

       TabJi = Table_ResultsAnalysed(Table_ResultsAnalysed.idxJ==countJ,:);
           % creates a subtable for countJ, i.e for a specific RPM cluster
           Jmean = mean(TabJi.J);
           tablesJ{countJ,1} = TabJi; % stores TableJi of  countJ loop
           tablesJ{countJ,2} =  Jmean;
        end
        % arranging in ascending order of Jmean
        [~,idJ] = sort([tablesJ{:,2}], 'ascend');
        tablesJ= tablesJ(idJ,:);

   end
%-------------------------------------------------------------------------

function [CT] = CT_calc(AoA, J, InterpFunc_wV_from_AoA_J)
    % From Eq. 19a AIAA paper or Eq. 3.22b in the thesis
    wV = InterpFunc_wV_from_AoA_J (AoA,J);
    CT = pi*J.^2/2*sqrt(wV.^2 + 2*wV.^3.*cosd(AoA)+wV.^4);
end

%-------------------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%     PLOT FUNCTIONS      %%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[f] = plotT_AoAo(obj)

    % 'TightInset' - expand the axis to fill figure complicated
    % set( gcf (figure to a size. 'Position', [ 8 8 10 8.5 ]
    %   is the export setup of width = 10cm height = 8.5 cm,
    %  8 8 is the position on screen - not important
    % set(gca, 'OuterPosition',[ 0.2 0 10 8.5 ]  is the axis position
    %   inside the figure.  0.2 puts better in the middle

            [RPM_Ao, V_Ao,T_Ao] = prepareSurfaceData(obj.RPM(obj.AoA==0),...
                      obj.V(obj.AoA==0),obj.T(obj.AoA==0));

             dummy = zeros(5,1);
             extraVs = [0 max(V_Ao)/4 max(V_Ao)/2 max(V_Ao)*0.75 max(V_Ao)]';

              RPM_Ao = [RPM_Ao;dummy] ;   % adding zero values of T, RPM
                                          % for the range of V_Ao
              V_Ao   = [V_Ao;extraVs] ;
              T_Ao   = [T_Ao;dummy] ;

              fitT_Ao_fromRPM_V = propeller.fitT_Ao (obj);

        f=    figure( 'Name', strcat('T x V,RPM [AoA=0] - ',obj.Name));
              plot( fitT_Ao_fromRPM_V, [RPM_Ao, V_Ao], T_Ao);

            set(gcf, 'units','centimeters','Position', [1 1 10.0 8.5]);
            set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

              set(gca,'FontName','TimesNewRoman','Fontsize',12,...
              'FontWeight','Normal','LineWidth',0.5,'layer','bottom',...
              'Box','on','LabelFontSizeMultiplier',1,'ZLim',[-0.,inf],...
              'XLim',[0,inf]);
              grid on
              xlabel ('RPM', 'FontSize', 12);
              ylabel ('V [m/s]', 'FontSize', 12);
              zlabel ('T [N]', 'FontSize', 12);

              view( -45, 25 );
              annotation('textbox',[0.1 0.85 0.345 0.0655],...
              'String',strcat(obj.Name, ' - AoA = 0\circ'),...
```

```
            'HorizontalAlignment','center','FontSize',10.,...
            'FitBoxToText','on','BackgroundColor',[1 1 1],...
            'FontName','TimesNewRoman','EdgeColor',[0. 0.45 0.74]);
    end
%-------------------------------------------------------------------------
function f =  plotCT_AoAo(obj)

            CT_Ao = obj.CT(obj.AoA==0);
            J_Ao = obj.J(obj.AoA==0);
            CT_Ao = CT_Ao(J_Ao>0);
            J_Ao=J_Ao(J_Ao>0);
            ft = propeller.fitCT_Ao(obj);

        f =  figure('Name',strcat('CT x J [AoA = 0] - ',obj.Name));

             h = plot( ft, J_Ao, CT_Ao );
                    % h is 2x1 array:  h(1)-> data,  h(2)-> fitline
            h(1).Marker = 'o';
            h(1).Color = [0 0 0];
            grid on;
            legend( '{C_T} vs J  (AoA=0)','FontSize', 10);
            ax = gca; %gca is current graph

          set(gcf, 'units','centimeters','Position', [1 3 10.0 8.5]);
          set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5],...
                'Fontsize', 12,'LabelFontSizeMultiplier',1);

            % Label axes
            xlabel ('J','FontSize', 12);
            ylabel ('C_T','FontSize', 12);
            grid on
            legend off

            annotation('textbox',[0.5 0.85 0.345 0.0655],...
            'String',strcat(obj.Name, ' - AoA = 0\circ'),...
            'HorizontalAlignment','center',...
            'FontSize',10,'FitBoxToText','on',...
            'EdgeColor',[0. 0.45 0.74],'BackgroundColor',[1 1 1]);
end
%-------------------------------------------------------------------------

function [ fits_wV_x_J, gofs_wV_x_J] = fits_wV_from_J_variousAs(tablesAoA)

    % FUNCTION CREATES FITS FOR wV X J FOR ALL tablesAoA OF THE PROPELLER
    % TESTED. FITS ONLY FOR THE AoAs IN tablesAoA
    % OUTPUT IS CELL ARRAY OF FITS FOR EVERY AoA.
    % INPUT OF THE FITS IS J. OUTPUT w/V
     warning 'off'

      szArray  = size(tablesAoA); % size of cell array that stores tables.
                                  % 2nd column stores AoA in each table
      maxcount   = szArray(1);   % number of rows, or number of total
                                      % tables stored in the array
      fits_wV_x_J = cell(maxcount,2);  % creates cell arrays (struct) to
                                     % store fitobjects in 1st column and
                                  % AoA in 2nd column (and to be plotted)
      gofs_wV_x_J = cell(maxcount,2);  % creates cell arrays to store gofs

            for k = 1:maxcount
               sizeTab = size(tablesAoA{k,1});  %size of each tableJ
                   if sizeTab(1,1) <= 3    % if height of each Table AoA,
                        % ie number of points to plot <=3, don't fit/plot
                     continue
                   end
               J  = tablesAoA{k}.J;
               wV = tablesAoA{k}.w_V;  %(tablesAoA{k}.w_V>0);
```

```
            %%{k,2} is the second column of
            %%the array of tables that stores the angle value
                  if tablesAoA{k,2}>60
                    kk = k;
                    break
                  end

            [x, y] = prepareCurveData( J, wV );  % removes Inf and NaN
            ft = fittype( 'a*x^-b+c' );
                        % function 'a*x^-b+c' for it intersects Xaxis
            opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
            opts.Display = 'Off';
            opts.StartPoint = [3 2 0.5 ];
            [fitobject, gof] = fit(x, y, ft, opts );

            fits_wV_x_J{k,1} = fitobject;  % fit equation ceofficients
                     % and object to be plotted. Stored at struct fits
            fits_wV_x_J{k,2} = tablesAoA{k,2};  % AoA value is stored
                     % in 2nd column of array tablesAoA. saved on 2nd
                     % column of fits_wV_x_J ->struct (maxcount,2)
            gofs_wV_x_J{k,1} = gof;  % goodness of fit stored at struct
                     % gofs with dimension (maxcount,1)
            gofs_wV_x_J{k,2} = tablesAoA{k,2};
          end
        %---------------------------------------
        %  FITS for AoA>60
          for k2 = kk:maxcount
            J2 = tablesAoA{k2}.J;     %(tablesAoA{k}.J>0); necessary to
                     % add if not using prepareCurveData(J,wV) below
            wV2 = tablesAoA{k2}.w_V;  %(tablesAoA{k}.J>0);
            [x2, y2] = prepareCurveData( J2, wV2 );  %removes INf and NaN

              opts2 = fitoptions( 'Method', 'NonlinearLeastSquares' );
            opts2.Display = 'Off';


          if tablesAoA{k2,2}>80
               ft2 = fittype( 'a*x^(-b)+c' );
           % ft2 = fittype( 'a*x^(-b)+ c' );
                 % POSSIBLE a*x^(-b) SOMETIMES better fit with this
                 % function for IT DOES NOT intersect xaxis
            opts2.StartPoint = [2 0.8 0.5];  % fit and plot very
                               % sensitive to Start point. obtained
                               % from curve fitting app->generate code
          elseif tablesAoA{k2,2}<=80
  %          ft2 = fittype( 'a*exp(-b*x^0.3)+d' );
             ft2 = fittype( 'a*x^(-b)+c' );
              % better fit with this function. IT DOES intersect xaxis
            opts2.StartPoint = [2 1 0.5];
          end

            [fitobject2, gof2] = fit(x2, y2, ft2, opts2 );

            fits_wV_x_J{k2,1} = fitobject2;  % fit equation ceofficients
                     % and object to be plotted. Stored at struct fits
            fits_wV_x_J{k2,2} = tablesAoA{k2,2};  % AoA value is stored
                     % in 2nd column of array tablesAoA. saved on 2nd
                     %  column of fits_wV_x_J ->struct (maxcount,2)
            gofs_wV_x_J{k2,1} = gof2;  % goodness of fit stored at struct
                     % gofs with dimension (maxcount,1)
            gofs_wV_x_J{k2,2} = tablesAoA{k2,2};
          end
      warning 'on'
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%-------------------------------------------------------------------------
    %  FUNCTION PLOTS w/V x J  3 FIGURES
    function [ InterpSurf_J_from_AoA_wV, InterpSurf_wV_from_AoA_J, ...
               figs_wVxJ_As] =     Plot_wV_vs_J_variousAoA(obj,tablesAoA)



      black = [0 0 0];
      lgtblu  = [0 0.7 1];
      cyan=[0 1 1];
      magenta = [1 0 1];
      yellow = [0.95 1 0]; % a bit darker
      blue = [0 0 1];
      green = [0 1 0];
      red = [1 0 0];
      purp = [0.62 0.14 0.72];
      orng = [1 0.55 0.13];
      drkgren =[0 0.25 0];
      oilgrn = [0 0.5 0.5];
      lemon = [0.8 0.95 0];
      turquoise = [0 0.75 0.65];
      brwn = [0.5 0.35 0.15];

      colormatrix = [red;blue;lgtblu;orng;green;brwn;magenta;oilgrn;...
                     blue;lemon;brwn;turquoise;purp;black];
      warning('off')  % avoid warning about fit ignoring NaN and Inf


      % ---------------------------------------------------------------
      %  1st figure
     figs_wVxJ_As(1) = figure ...
               ('Name',strcat('wV x J [many AoA LE 60] - ',obj.Name));
      hold on;
      grid

      szArray  = size(tablesAoA); % size of cell array that stores tables.
                                  % 2nd column stores AoA in each table
      maxcount   = szArray(1);   % number of rows, or number of total

      %  CALLING FIT FUNCTION FOR w/V given J
      fits_wV_x_J = propeller.fits_wV_from_J_variousAs(tablesAoA) ;

          for k = 1:maxcount    % loop for every value of AoA
               sizeTab = size(tablesAoA{k,1});  %size of each tableJ
                   if sizeTab(1,1) <= 3    % if height of each Table AoA,
                        % ie number of points to plot <=3, don't fit/plot
                     continue
                   end

                   if tablesAoA{k,2}>60
                   %%{k,2} is the second column of
                   %% the array of tables that stores the angle value

                      kk = k;
                      break  % for AoA>60 , another figure below
                   end

               xinterval = linspace(0,1.5);    % x interval for the plot
               h1  = fits_wV_x_J{k}(xinterval'); % y = w/V given J

               plot (xinterval,h1,'Color',colormatrix(k,:),...
                         'HandleVisibility','callback','LineWidth',1);

               J  = tablesAoA{k}.J;
               wV = tablesAoA{k}.w_V;  %(tablesAoA{k}.w_V>0);
               [x1, y1] = prepareCurveData(J,wV);   % cleans NaN Inf
```

```
            legvalue = num2str(tablesAoA{k,2});  %converting MeanJ into
               % string with 2 decimal places precision only for legend

            scatter(x1,y1,'filled','DisplayName',...
                strcat('AoA= ',legvalue,'\circ'),'MarkerEdgeColor',...
                '[0 0 0]','MarkerFaceColor',colormatrix(k,:),...
                'LineWidth',0.7,'SizeData',12, 'HandleVisibility','on');


        end
            % plotting w/V = 0
            yline(0,'- k','HandleVisibility','callback','LineWidth',1.);

        set(gcf, 'units','centimeters','Position', [1 12 10.0 8.5]);
        set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);
        set(gca, 'FontName','TimesNewRoman','fontsize',12,'xminorgrid',...
           'on','yminorgrid','on','XMinorTick','on','YMinorTick','off',...
              'LabelFontSizeMultiplier',1);

            xlabel ('J','fontsize',12);ylabel ('w/V','fontsize',12)
            xlim ([0.2 1.1])
            ylim([-0.05 0.6])
            legend off  %turning off and on later. bug on legend
            legend('Position',[0.648 0.67 0.26 0.165],'Autoupdate',...
                'on','fontsize',...
                9,'FontName','TimesNewRoman','EdgeColor',[0.8 0.8 0.8])

             annotation('textbox',[0.32 0.82 0.345 0.0655],...
            'String',obj.Name,'FontName','TimesNewRoman',...
            'HorizontalAlignment','center',...
            'FontSize',9,'FitBoxToText','on',...
            'EdgeColor',[0. 0.45 0.74],'BackgroundColor',[1 1 1]);

            str = {'fit type: $$ax^{-b}+c$$'};
            annotation('textbox',[0.59 0.52 0.345 0.0655],...
            'String',str,'FontName','TimesNewRoman',...
            'HorizontalAlignment','center','interpreter','latex',...
            'FontSize',9,'FitBoxToText','on',...
            'EdgeColor',[0.8 0.8 0.8],'BackgroundColor',[1 1 1]);




% ------------------------------------------------------------------
%  2nd figure
figs_wVxJ_As(2) = ...
     figure ('Name',strcat('wV x J [many AoA GT 60] - ',obj.Name));

hold on;
grid
   for k2 = kk:maxcount

            xinterval = linspace(0,1.3);    % x interval for the plot

              sizeTab = size(tablesAoA{k2,1});  %size of each tableJ
                if sizeTab(1,1) <= 3    % if height of each Table AoA,
                      % ie number of points to plot <=3, don't fit/plot
                  continue
                end

            y2  = fits_wV_x_J{k2}(xinterval'); % y = w/V given J

            plot (xinterval,y2,'Color',colormatrix(k2,:),...
                      'HandleVisibility','callback','LineWidth',1);
```

```
                J  = tablesAoA{k2}.J;
                wV = tablesAoA{k2}.w_V;  %(tablesAoA{k}.w_V>0);
                [x2, y2] = prepareCurveData(J,wV);    % cleans NaN Inf

                legvalue = num2str(tablesAoA{k2,2});  %converting AoA value
                       % for legend (stored in the 2nd column of the struct)
                scatter(x2,y2,'filled','DisplayName',...
                    strcat('AoA= ',legvalue,'\circ'),'MarkerEdgeColor',...
                    '[0 0 0]','MarkerFaceColor',colormatrix(k2,:),...
                    'LineWidth',1,'SizeData',12, 'HandleVisibility','on');

            end
                % plotting w/V = 0
                yline(0,'- k','HandleVisibility','callback','LineWidth',1.);

             set(gcf, 'units','centimeters','Position', [1 18 10.0 8.5]);
             set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);
             set(gca,'FontName','TimesNewRoman','FontSize',12,...
                        'XMinorTick','on','YMinorTick','off',...
                         'xminorgrid','on','yminorgrid','on',...
                         'LabelFontSizeMultiplier',1);

                xlabel ('J','fontsize',12);   ylabel ('w/V','fontsize',12);
                xlim ([0.2 1.3]);
                ylim([-0.05 0.9]);
                legend off % turning off and on later. bug on legend Matlab
                legend('Position',[0.65 0.72 0.25 0.165],'Autoupdate','on',...
                                'fontsize', 9,'EdgeColor',[0.8 0.8 0.8])
                 annotation('textbox',[0.32 0.82 0.345 0.0655],...
                'String',obj.Name,'FontSize',9,'FitBoxToText','on',...
                'HorizontalAlignment','center','EdgeColor',[0. 0.45 0.74],...
                'BackgroundColor',[1 1 1],'FontName','TimesNewRoman');

%                 str = {'fit types:',...
%                         'for $$(\alpha_p=90^\circ)$$: $$ax^{-b}$$' ,...
%                         'else: $$ax^{-b}+c$$'};

                str = {'fit type: $$ax^{-b}+c$$'};

                 annotation('textbox',[0.62 0.55 0.345 0.0655],...
                'String',str,'FontSize',9,'FitBoxToText','on',...
                'interpreter','latex','HorizontalAlignment','left',...
                'EdgeColor',[0.8 0.8 0.8],'BackgroundColor',[1 1 1],...
                'FontName','TimesNewRoman');
        % ------------------------------------------------------------------
        %   3rd figure
        %   CREATING INTERPOLATION SURFACE BASED ON scattered data and added
        %   points from extrapolation of each AoA fit. (more accurate)
        %   Functions scatteredInterpolant (extrapolates outside scatter data)
        %   or griddata (only interpolates within data)

            J_points  = (0.1:0.1:1.4)'; % __ points
            len       = length(J_points); %
            MatData   = [];
          % MatData   = zeros(len*maxcount,3);
          for k3 = 1:maxcount          %loop for all AoA values in tablesAoA

              sizeTab = size(tablesAoA{k3,1});  %size of each tableJ
                    if sizeTab(1,1) <= 3    % if height of each Table AoA,
                         % ie number of points to plot <=3, don't fit/plot
                      continue
                    end

              w_Vpoints  = fits_wV_x_J{k3,1}(J_points);  % equivalent w/V
                           % calculated from J_points based on each AoA fit
                AoA_points = repmat(tablesAoA{k3,2},len,1);
                    % creates a vector with repeated values tablesAoA{k,2}
```

```
                % with length len and 1 column
            MatData     = [MatData; [AoA_points w_Vpoints J_points]];
                % creates a matrix for data
        end

          AoApts = MatData(:,1);
          wVpts = MatData(:,2);
          Jpts = MatData(:,3);
        %-------------------------------------------------------------------
          InterpSurf_J_from_AoA_wV = scatteredInterpolant...
                                                (AoApts,wVpts,Jpts);
                                    % calculates interpolant function
          [X, Y] = meshgrid(0:10:90,-0.1:0.1:3.5);
                                      % defines a uniform grid of X, Y

          Z = InterpSurf_J_from_AoA_wV(X,Y)    ;
              % calculates and plots  the fit from function F over unifrom
              % meshgrid (X,Y) are input AoA and w/V -> output is J

        %-------------------------------------------------------------------
           InterpSurf_wV_from_AoA_J = scatteredInterpolant...
                                                (AoApts,Jpts,wVpts);
                                    % calculates interpolant function
          [XX, YY] = meshgrid(0:10:90,0.1:0.1:3.5);
                                      % defines a uniform grid of X, Y

          InterpSurf_wV_from_AoA_J(XX,YY)    ;
              % calculates and plots the fit from function F over uniform
              % meshgrid (X,Y) are input AoA and w/V -> output is J
        %-------------------------------------------------------------------
figs_wVxJ_As(3) = ...
                  figure('Name',strcat(...
                          'Interp_SURF J from AoA and wV - ',obj.Name));

          scatter3(AoApts,wVpts,Jpts,'filled','MarkerFaceColor',...
              [1 0.41 0.16],'MarkerEdgeColor','k','LineWidth',0.7,...
              'SizeData',10);
          hold on
                              % mesh(X,Y,Z)  % option to surf (creates 3d
                              % surface mesh also creates surface but
                              % different appearance
          surf(X,Y,Z,'FaceColor','interp','FaceAlpha',0.4,'EdgeColor',...
                                                    [0.15 0.15 0.15])
          zlim([0 1.6]);     xlim([0 90]);      ylim([-0.2 1.5]);
          vectorXticks = 0:10:90;
          xticks(vectorXticks);
          xticklabels({'0','','','30','','','60','','','90'});
          vectorYticks = 0:0.3:1.5;
          yticks(vectorYticks);

      set(gcf, 'units','centimeters','Position', [1 17 10.0 8.5]);
      set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);
          set(gca,'FontName','TimesNewRoman','FontSize',12,...
                    'XMinorTick','off','YMinorTick','off',...
                    'LabelFontSizeMultiplier',1);
          xlabel('AoA [deg]','fontsize',12); ylabel ('w/V','fontsize',12);
          zlabel ('J','fontsize',12)

          annotation('textbox',[0.4 0.9 0.345 0.0655],...
               'String',obj.Name,...
               'HorizontalAlignment','center',...
               'FontSize',10,'FitBoxToText','on',...
               'EdgeColor',[0. 0.45 0.74],'BackgroundColor',[1 1 1]);
          view(120,20)
          warning('on')  % turning warning back on

      end
```

```
%-------------------------------------------------------------------------

function f =  Plot_TwTax_vs_wV_variousAoA(obj, tablesAoA)

    szArray  = size(tablesAoA); % size of cell array that stores tables.
                        % 2nd column stores average of J in each table
    maxcount   = szArray(1);    % number of rows, or number of total
                                % tables stored in the array. creating
                                % cell arrays to store scatter objects.
    % NO NEED FOR FITS HERE. Just plots of theoretical w/V according to
    % equation Twing/Taxial

    scatTwTa_vs_wV     = cell(maxcount,1);

    black = [0 0 0];
    lgtblu  = [0 0.7 1];
    cyan=[0 1 1];
    magenta = [1 0 1];
    yellow = [0.95 1 0]; % a bit darker
    blue = [0 0 1];
    green = [0 1 0];
    red = [1 0 0];
    purp = [0.62 0.14 0.72];
    orng = [1 0.55 0.13];
    drkgrn =[0 0.25 0];
    oilgrn = [0 0.5 0.5];
    lemon1 = [0.6 0.95 0];
    lemon2 = [0.8 0.95 0];
    turquoise = [0 0.75 0.65];
    brwn = [0.5 0.35 0.15];

    colormatrix = [red;blue;lgtblu;orng;green;brwn;magenta;blue;...
           brwn;turquoise;lemon1;black;yellow;lemon2;cyan;purp;drkgrn];

   f = figure ('Name',strcat('Tw_Tax X wV [many AoA] - ',obj.Name));
    hold on;
    grid

        for k = 1:maxcount
           sizeTab = size(tablesAoA{k,1});    %size of each tableAoA
           if sizeTab(1,1) <= 3        % if height of each Table AoA,
                   % ie number of points to plot <=3, don't fit/plot
                continue
           end

            wV = tablesAoA{k}.w_V;  %(tablesAoA{k}.J>0);
            TwTa = tablesAoA{k}.Twing_Taxial; %(tablesAoA{k}.J>0);

            ang = tablesAoA{k}.AoA(1);   % captures only first of all
                % equal AoAs in every table. For function calculation
            wVinterval = linspace(0,1.3);   % x interval for the plot
            y = propeller.TwTax_from_wV (ang, wVinterval);
                % NOT FIT. Uses equation from propeller.Twing_over_Taxial
            plot (wVinterval,y,'Color',colormatrix(k,:),...
                        'HandleVisibility','callback','LineWidth',1.0);

            legvalue = num2str(tablesAoA{k,2});  %converting MeanJ into
               % string with 2 decimal places precision only for legend
            scatTwTa_vs_wV{k} = scatter(wV,TwTa,'filled','DisplayName',...
                strcat('AoA= ',legvalue,'\circ'),'MarkerEdgeColor',...
                '[0 0 0]','MarkerFaceColor',colormatrix(k,:),...
                'LineWidth',1.0,'SizeData',14, 'HandleVisibility','on');
        end

   yline(1,'-.r','HandleVisibility','callback');
                               % plotting limit of Twing = Taxial
   xline(0.57735, '-.r','HandleVisibility','callback');
```

```
                                % plotting w limit for Twing = Taxial

    set(gcf, 'units','centimeters','Position', [12 1 10.0 8.5]);
    set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

    set(gca,'FontName','timesNewRoman','FontSize',12,'xlim',[0 1.1],...
        'xtick',[0 0.2 0.4 0.6 0.8 1.0 ], 'yscale','log',...
        'ylim',[0.01 50],'ytick',[0.01, 0.1,1,10,50],...
          'XMinorTick','on','YMinorTick','on');

    annotation('textarrow',[0.3,0.3],[0.3 0.3],'Units', 'normalized',...
     'FontName','timesNewRoman','Position',[0.664 0.23 -0.07 0],...
                'String','w/V = 0.57735','FontSize', 10,'Linewidth',1);
    annotation('textbox',[0.34 0.83 0.345 0.0655],'String',obj.Name,...
             'HorizontalAlignment','center','FontSize',9,'FitBoxToText',...
             'on','EdgeColor',[0. 0.45 0.74],'BackgroundColor',[1 1 1],...
             'FontName','timesNewRoman');

             % cause theoretical value of Tw/Ta(AoA=90, w/V=0) is 4.7588
    xlabel ('w / V','fontsize',12);
    ylabel ('T_{wing}/T_{axial}','fontsize',12)
    legend off  %turning off and on later. bug on legend
    legend('Position',[0.672 0.547 0.24 0.3723],'Autoupdate','on',...
                            'fontsize',8,'EdgeColor','[0.8 0.8 0.8]')
end


%-------------------------------------------------------------------------

function [ fig_TwTa_x_J_As ] = ...
                            Plot_TwTax_vs_J_variousAoA(obj, tablesAoA)

    szArray  = size(tablesAoA); % size of cell array that stores tables.
                        % 2nd column stores average of J in each table
    maxcount   = szArray(1);    % number of rows, or number of total
                            % tables stored in the array


    black = [0 0 0];
    lgtblu  = [0 0.7 1];
    cyan=[0 1 1];
    magenta = [1 0 1];
    yellow = [0.95 1 0]; % a bit darker
    blue = [0 0 1];
    green = [0 1 0];
    red = [1 0 0];
    purp = [0.62 0.14 0.72];
    orng = [1 0.55 0.13];
    drkgren =[0 0.25 0];
    oilgrn = [0 0.5 0.5];
    lemon1 = [0.6 0.95 0];
    lemon2 = [0.8 0.95 0];
    turquoise = [0 0.75 0.65];
    brwn = [0.5 0.35 0.15];

    colormatrix = [red;blue;lgtblu;orng;green;brwn;magenta;blue;...
                                    lemon2;brwn;turquoise;lemon1;black];
%-------------------------------------------------------------------------
   fig_TwTa_x_J_As(1) = figure ('Name',...
                       strcat('Tw_Tax X J [many AoA LE 60] - ',obj.Name));

    hold on;
    grid
        ax = yline(0,'-','HandleVisibility','callback','LineWidth',1.5);
                        %plotting a line for Aoa=0 cause fit function
        ax.Color =red;
        opts = fitoptions('Normalize','on'); % does not converge with all
```

```
                                        % values=0
for k = 1:maxcount

    sizeTab = size(tablesAoA{k,1});  %size of each tableAoA
        if sizeTab(1,1) <= 3     % if height of each Table AoA,ie number
                                 % of points to plot <=3, don't fit/plot
         continue
        end

    if tablesAoA{k,2}>60    %%{k,2} is the second column of the array of
                            %% tables that stores the angle value

        kk = k;
        break
    end

 Jinterval = linspace(0,1.2);        % x interval for the plot

  [fits_wV_x_J] = ...
                      propeller.fits_wV_from_J_variousAs(tablesAoA);
 % retrieving the fits for every AoA of tablesAoA
 % BETTER TO USE THESE FITS THAN SCATTTERED INTERPOLANT FOR THE SURFACES.
 wV_equiv = fits_wV_x_J{k}(Jinterval);  % calculating w/V for all J
 AoA      = fits_wV_x_J{k,2};   %gathering wach aoA value

 sz    = size(Jinterval);  % finding the size of horiz vector Jinterval
 leng  = sz(2);            % 2nd column of size is length of vector
 AoAvect  = repmat(AoA,leng,1);  % craeating aaoa vector of same values
                                 % of same length of Jinterval
 TwTa     = propeller.TwTax_from_wV (AoAvect, wV_equiv);
                            % calculating exact value of TwTa from wV
                            % equiv to J from wV x J fits

  plot (Jinterval,TwTa,'Color',colormatrix(k,:),'HandleVisibility',...
                                          'callback','LineWidth',1);

  dummyzero = zeros(maxcount,2);
  tabledummy = array2table(dummyzero,'Variablenames',{'J','TwTa'});
  Jscatter  = [tabledummy.J; tablesAoA{k}.J];% adding data of J=0->Tw_Ta=0
  TwTa_scatter = [tabledummy.TwTa;tablesAoA{k}.Twing_Taxial];

  legvalue = num2str(tablesAoA{k,2});  % converting AoA value into
                                       % string  for legend
  scatter(Jscatter,TwTa_scatter,'filled','MarkerEdgeColor','[0 0 0]',...
            'DisplayName',strcat('AoA= ',legvalue,'\circ'),...
            'MarkerFaceColor',colormatrix(k,:),'LineWidth',1.0,...
            'SizeData',16, 'HandleVisibility','on');

end

    yline(1,'--r','HandleVisibility','callback','linewidth',1,...
                                                  'alpha',1);
                % plotting limit of Twing = Taxial

   set(gcf, 'units','centimeters','Position', [12 11 10.0 8.5]);
   set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);
    set(gca,'FontName','TimesNewroman','FontSize',12,'ylim',[-0.01 1.2],...
     'xlim', [0 round(1.3*max(Jscatter))],'XMinorTick','on',...
     'YMinorTick','on');

    xlabel ('J','FontName','TimesNewroman','FontSize',12);
    ylabel ('T_{wing}/T_{axial}','FontName','TimesNewroman','FontSize',12)
    legend off  %turning off and on later. bug on legend
    legend('Location','northwest','Autoupdate','on','fontsize',9,...
                  'EdgeColor',[0.8 0.8 0.8])
    annotation('textbox',[0.48 0.83 0.345 0.0655],'String',obj.Name,...
        'HorizontalAlignment','center','FontSize',10,'FitBoxToText',...
```

```
            'on','EdgeColor',[0. 0.45 0.74],'BackgroundColor',[1 1 1],...
             'FontName','TimesNewRoman');
        hold off
%-------------------------------------------------------------------------
%   2nd FIGURE
    fig_TwTa_x_J_As(2) = figure ('Name',...
                         strcat('Tw_Tax X J [many AoA GT 60] - ',obj.Name));
    hold on;
    grid
    for k2 = kk:maxcount

     wV_equiv = fits_wV_x_J{k2}(Jinterval);  % calculating w/V for all J
     AoA      = fits_wV_x_J{k2,2};

     sz    = size(Jinterval);
     leng  = sz(2);
     AoAvect  = repmat(AoA,leng,1);
     TwTa     = propeller.TwTax_from_wV (AoAvect, wV_equiv);

        plot (Jinterval,TwTa,'Color',colormatrix(k2,:),'HandleVisibility',...
                                              'callback','LineWidth',1.0);

        Jscatter = [tabledummy.J; tablesAoA{k2}.J];  % adding data of J=0 ->Tw_Ta=0
        TwTa_scatter = [tabledummy.TwTa;tablesAoA{k2}.Twing_Taxial];

        legvalue = num2str(tablesAoA{k2,2});  % converting AoA value into
                                             % string  for legend
        scatter(Jscatter,TwTa_scatter,'filled','LineWidth',1.0,'DisplayName',...
            strcat('AoA= ',legvalue,'\circ'),'MarkerEdgeColor','[0 0 0]',...
                  'MarkerFaceColor',colormatrix(k2,:),'SizeData',16,...
                  'HandleVisibility','on');
    end

        yline(1,'--r','HandleVisibility','callback','linewidth',1,...
                                                    'alpha',1);
                % plotting limit of Twing = Taxial
       set(gcf, 'units','centimeters','Position', [12 18 10.0 8.5]);
       set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

        set(gca,'FontName','TimesNewroman','FontSize',12,'ylim',[0 6],...
         'xlim',[0 round(1.25*max(Jscatter))],'XMinorTick','on',...
          'YMinorTick','on');

        xlabel ('J','FontName','TimesNewroman','FontSize',12)
        ylabel ('T_{wing}/T_{axial}','FontName','TimesNewroman','FontSize',12)
        legend off  %turning off and on later. bug on legend
        legend('Location','northwest','Autoupdate','on','fontsize',9,...
                  'FontName','TimesNewRoman',  'EdgeColor',[0.8 0.8 0.8])
        annotation('textbox',[0.48 0.83 0.345 0.0655],'String',obj.Name,...
            'HorizontalAlignment','center','FontSize',10,'FitBoxToText',...
            'on','EdgeColor',[0. 0.45 0.74],'BackgroundColor',[1 1 1],...
             'FontName','TimesNewRoman');
    end


  %-----------------------------------------------------------------------

       function [fitsCTxJ, gofs_CTxJ, figCTxJ_As] = ...
                                        Plot_CTxJ_variousAoA(obj,tablesAoA)
        % OTHER FUNCTION FROM SURF PLOT AND FIT OF ALL DATA IS GLOBAL AND
        % USED IN THE THESIS
        % function fits fit(poly2) BETTER THAN polyfit2() to the data.
        % Outliers distort fits. Run plot and delete outlier data from row
        % at specific tableAoA for that angle.   Run the plot again.

        szArray  = size(tablesAoA); % size of cell array that stores tables.
                           % 2nd column stores average of J in each table
```

```
    maxcount    = szArray(1);    % number of rows, or number of total
                                 % tables stored in the array
    fitsCTxJ = cell(maxcount,2);   % creates cell arrays to store
                                 % fitobjects,(to be plotted).
                                 % 2nd colums is value of AoA
    gofs_CTxJ = cell(maxcount,2);

    figCTxJ_As = figure ('Name',...
                 strcat('CT X J [many AoA] - ',obj.Name));
    hold on;
    grid on

    black = [0 0 0];
    lgtblu  = [0 0.7 1];
    cyan=[0 1 1];
    magenta = [1 0 1];
    yellow = [0.95 1 0]; % a bit darker
    blue = [0 0 1];
    green = [0 1 0];
    red = [1 0 0];
    purp = [0.62 0.14 0.72];
    orng = [1 0.55 0.13];
    drkgren =[0 0.25 0];
    oilgrn = [0 0.5 0.5];
    lemon = [0.8 0.95 0];
    turquoise = [0 0.75 0.65];
    brwn = [0.5 0.35 0.15];
    colormatrix = [red;blue;lgtblu;orng;green;brwn;magenta;yellow;blue;...
                                        red;turquoise;lemon;black];

    CTJo = mean(tablesAoA{1}.CT(tablesAoA{1}.J==0));
      % defining an average value at V=0 for all CT measured at
      % different RPMs. Conceptually wrong, but using this average value
      % just as a reference to smooth the CT fits x J. The fits will be
      % closer to reality. Many plots diverge completely without this

   for k = 1:maxcount
          sizeTab = size(tablesAoA{k,1});  %size of each tableJ
           if sizeTab(1,1) <= 3 % is height of each Table AoA,
                     % ie number of points to plot <=3, don't fit/plot
                continue
           end
           if (tablesAoA{k,2}>=60)
            % defining a CTJo point only for better fit curve at
            % AoA>=60 as being the same as the CT of the lowest J
            % measured.
%            CTJo = tablesAoA{k}.CT(tablesAoA{k}.J==min(tablesAoA{k}.J));
           end

          xx = tablesAoA{k}.J(tablesAoA{k}.J>0);
          yy = tablesAoA{k}.CT(tablesAoA{k}.J>0);
          x1 = [0;xx];
          y1 = [CTJo;yy];

         [fitobject1, gof1] = fit(x1,y1,'poly2');

      % saving both fit and gof into cell array fitsCTxJ
      fitsCTxJ{k,1} = fitobject1;  % fit equation ceofficients and
                                   % object to be plotted
      fitsCTxJ{k,2} = tablesAoA{k}.AoA(1); % saving only first element
                       % of vector AoA in each case (all equal values)
      gofs_CTxJ{k,1} = gof1;        % goodness of fit
      gofs_CTxJ{k,2} = tablesAoA{k}.AoA(1);% saving only first element
                       % of vector AoA in each case (all equal values)

          xinterval = linspace(0,1.3);      % x interval for the plot
          h1 = feval(fitobject1,xinterval);  % evaluates the fit within
```

```
            % xinterval otherwise matlab wil plot only between data limit
        plot (xinterval,h1,'Color',colormatrix(k,:),'LineWidth',1....
                                ,'HandleVisibility','callback');
            legvalue = num2str(tablesAoA{k,2});  %converting MeanJ
            % into string with 2 decimal places precision only for legend

        scatter(x1,y1,'filled','DisplayName',...
                    strcat('AoA= ',legvalue,'\circ'),'MarkerEdgeColor',...
                  [0 0 0],'MarkerFaceColor',colormatrix(k,:),...
                 'SizeData',12, 'HandleVisibility','on','LineWidth',0.3);
    end %loop

     set(gcf, 'units','centimeters','Position', [22 1 10.0 8.5]);
     set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

      set(gca,'FontSize',12,'YLim',[0 1.5*max(y1)],...
          'XLim',[0.15 round(1.1*max(x1))],'FontName','TimesNewRoman')

      xlabel ('J','fontsize',12);ylabel ('C_T','fontsize',12)
      legend off  %turning off and on later. bug on legend
      legend('Location','northwest','Autoupdate','on','fontsize',8,...
             'FontName','TimesNewRoman', 'EdgeColor',[0.8 0.80 0.80])
      annotation('textbox',[0.45 0.82 0.345 0.0655],...
              'String',obj.Name,...
              'HorizontalAlignment','center',...
              'FontSize',9,'FontName','TimesNewRoman','FitBoxToText','on',...
              'EdgeColor',[0. 0.45 0.74],'BackgroundColor',[1 1 1]);
      end

%_____________________________________________________________________________
%%%%%%%%%%%%%%%%      VERY IMPORTANT          %%%%%%%%%%%%%%%%%%%%
      function [fitCTxA_J_Surf, gofCTxA_J, fig_Surf_CTxAxJ, ...
                                                  fig_Mesh_CTxAxJ ] = ...
                Plot_Surf_CTxAxJ(obj, InterpSurf_wV_from_AoA_J, type_fit)

%%%%%%%%%%%%%%%%      VERY IMPORTANT          %%%%%%%%%%%%%%%%%%%%
      %           from this interpolated surface stems
      %    the other plots CTxJ many AoAs and CT x AoA many Js
      AoA = obj.AoA;
      J = obj.J;
      CT = obj.CT;

      [xData, yData, zData] = prepareSurfaceData( AoA, J, CT );
              % clear data of NaN and Inf

%         type =  'poly44';
      ft = fittype( type_fit );

      % Fit model to data.
      [fitCTxA_J_Surf, gofCTxA_J] = fit( [xData, yData], zData, ft, ...
                                                  'Normalize', 'on');
      % Plot fit with data.
      %-------------------------------------------------------
      % 1ST FIGURE
     fig_Surf_CTxAxJ =  figure( 'Name', ...
                strcat('SURF CT x (AoA, J) - ', obj.Name),'Colormap',pink);

      plot( fitCTxA_J_Surf, [xData, yData], zData );

          vectorXticks = 0:15:90;
          xticks(vectorXticks);
          vectorYticks = 0.1:0.1:1.1;
          yticks(vectorYticks);

     set(gcf, 'units','centimeters','Position', [22 12 10.0 8.5]);
     set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);
```

```
        set(gca,'FontName','TimesNewRoman','Fontsize',12,...
            'LabelFontSizeMultiplier',1,...
            'YTickLabel',{'','0.2','','0.4','','0.6','','0.8','','1.0',''},...
            'XTickLabel',{'0','','30','','60','','90'},'Box','off');

        annotation('textbox',[0.35 0.87 0.345 0.0655],'String',obj.Name,...
              'HorizontalAlignment','center','FontName','TimesNewRoman',...
              'FontSize',9,'FitBoxToText','on','EdgeColor',[0. 0.45 0.74],...
              'BackgroundColor',[1 1 1]);

%       str = {'fit: ',type ,'for $AoA\rightarrow (4)$'...
%                                          ,'for $J\rightarrow (2)$'};
        str = {'fit: ',type_fit ,'$(AoA, J)$'};
%

        annotation('textbox',[0.72 0.87 0.45 0.0655],...
              'String',str,'FontSize',8,'FitBoxToText','on',...
              'interpreter','latex','HorizontalAlignment','left',...
              'EdgeColor',[0.8 0.8 0.8],'BackgroundColor',[1 1 1]);
        ylim ([0.1 1.1*max(yData)]);
        zlim ([-0.01 1.1*max(zData)]);
        % Label axes
        xlabel 'AoA [deg]'
        ylabel J
        zlabel C_T
        grid on
        %view( -116.0, 18 );
        view( -125.0, 20 );
        %-------------------------------------------------------------
         % 2ND FIGURE
        fig_Mesh_CTxAxJ = figure('Name',...
            strcat('MESH CT x (AoA, J) - ',obj.Name),'Colormap',summer);

        % creating a mesh with AoA, J values
        [Amesh, Jmesh] = meshgrid(0:5:90 , 0:0.1:1.1);
        CTmesh = fitCTxA_J_Surf(Amesh,Jmesh);

        surf(Amesh,Jmesh,CTmesh,'FaceColor','interp','FaceAlpha',...
                                    0.2,'EdgeColor',[0.15 0.15 0.15]);
        hold on

        MatrixTwTax = propeller.TwTax_from_J (Amesh, Jmesh, ...
                                              InterpSurf_wV_from_AoA_J);
        den=1+MatrixTwTax;
        CTax_mesh = CTmesh./den;
        CTwing_mesh = CTmesh - CTax_mesh;

        surf(Amesh,Jmesh,CTax_mesh,'FaceColor',[0.93 0.58 0.58],...
                        'FaceAlpha',0.7,'EdgeColor',[0.15 0.15 0.15]);

        surf(Amesh,Jmesh,CTwing_mesh,'FaceColor',[0.45 0.74 0.93],...
                           'FaceAlpha',0.7,'EdgeColor',[0.15 0.15 0.15]);

        zlim ([-0.00 1.1*max(zData)]);
        xlim ([0 90]);
        vectorXticks1 = 0:10:90;
        xticks(vectorXticks1);
        yticks(0.1:0.1:1.1);

       set(gcf, 'units','centimeters','Position', [22 16 10.0 8.5]);
       set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

        set(gca,'FontName','TimesNewRoman','Fontsize',12,'Box','off',...
            'LabelFontSizeMultiplier',1,'XMinortick','off','YMinorGrid',...
            'off','XTickLabel',{'0','','','30','','','60','','','90'},...
            'YTickLabel',{'','0.2','','0.4','','0.6','','0.8','','1.0',''},...
            'View',[44 23]);
```

```
        legend('C_T','C_T_{axial}','C_T_{wing}','FontName','TimesNewRoman',...
            'Location','northeast','Autoupdate','on','fontsize',8,...
            'EdgeColor','[0.8 0.8 0.8]');

        annotation('textbox',[0.35 0.87 0.345 0.0655],'String',obj.Name,...
              'HorizontalAlignment','center','FontName','TimesNewRoman',...
              'FontSize',9,'FitBoxToText','on','EdgeColor',[0. 0.45 0.74],...
              'BackgroundColor',[1 1 1]);

        xlabel('AoA [deg]') ;
        ylabel('J') ;
        zlabel('C_T');
        %view( -116.0, 18 );
        view( -125.0, 20 );

        end
%-------------------------------------------------------------------------
%
function [figsCT_CTa_CTw_xJ, figCTsxJ, figCToverJxJ] = ...
                        Plot_CTs_x_J_variosA_fromfit(obj,fitCTxA_J_Surf ,...
                      InterpSurf_wV_from_AoA_J,maxJ, optionalHorzVectorAoAs)

switch nargin   % allows for variable number of input arguments
    case 4      %  if last argument is not provided code chooses many AoAs
        Alow= 0:15:60;  % horizontal vector
        Ahigh = 65:5:90; % horizontal vector
        A=[Alow Ahigh];
    case 5
        A = optionalHorzVectorAoAs; % MUST BE horizontal vector
    otherwise
        disp 'Wrong number of input arguments'
end

        black = [0 0 0];
        lgtblu  = [0 0.7 1];
        cyan=[0 1 1];
        magenta = [1 0 1];
        yellow = [0.95 1 0]; % a bit darker
        blue = [0 0 1];
        green = [0 1 0];
        red = [1 0 0];
        purp = [0.62 0.14 0.72];
        orng = [1 0.55 0.13];
        drkgren =[0 0.25 0];
        oilgrn = [0 0.5 0.5];
        lemon = [0.8 0.95 0];
        turquoise = [0 0.75 0.65];
        brwn = [0.5 0.35 0.15];

        colormatrix = [red;blue;lgtblu;orng;green;oilgrn;brwn;magenta;...
                        yellow;cyan;purp;magenta;turquoise;black];

    J =( 0.1:0.1:maxJ)'; % vertical vector SET LIMIT
    len = length(J);

    ang = repmat(A,length(J),1); % repeats the matrix A length(J) times in
                                 % number rows and 1 time the number os columns

    dimension = size(ang); % returns length and width
    wide = dimension(2); % calls only width
    figsCT_CTa_CTw_xJ = gobjects(1,wide);% create an array(wideX1)of figures

  for k = 1:wide

    % ang(1,k);  % taking only the first element of the vector of AoA which
    % all values are the same. Length to match length of J
```

```
    [CT, CTax, CTwing] = FUN_CTax_CTwing_from_AoA_J(ang(:,k), J,...
                                fitCTxA_J_Surf ,InterpSurf_wV_from_AoA_J);

    %-------------------------------------------------------------
    % 1ST SET OF FIGURES CT, CTaxial, CTwing x J FOR ASSORTED AoAs

 %   figsCT_CTa_CTw_xJ = gobjects(wide,1); % preallocate graphic objects array
    figsCT_CTa_CTw_xJ(k) = figure ('Name',...
            strcat(obj.Name,' - CT, CTax, CTw x J - AoA=',string(ang(1,k))));
        grid on
        hold on

    plot(J,CT,'Color',oilgrn,'LineWidth',1)
    plot(J,CTax,'LineWidth',1)
    plot(J,CTwing,'Color',[0.53 0.78 0.95],'LineWidth',1)

    set(gcf, 'units','centimeters','Position', [33 1 10.0 8.5]);
    set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

    set(gca,'FontName','TimesNewRoman','FontSize',12,...
            'xminorgrid','on','yminorgrid','on','LabelFontSizeMultiplier',1)
    annotation('textbox',[0.22 0.71 0.3 0.2],'FitBoxToText','on','string',...
    strcat(obj.Name,' - AoA =',string(ang(1,k)),'\circ'),'BackgroundColor',...
    [1 1 1],'EdgeColor',[0. 0.45 0.74],'FontName','TimesNewRoman',...
    'FontSize',9);
    legend('C_T','C_T_{axial}','C_T_{wing}','EdgeColor',[0.8 0.8 0.8],...
                                    'FontName','TimesNewRoman','FontSize',8)
    xlabel('J')
    ylabel('C_T')
    ylim([0 1.4*max(CT)]);
    xlim([0.1 maxJ]);
  end
%-------------------------------------------------------------
    % 2ND FIGURE CT x J FOR ASSORTED AoAs
    dispname = strings(wide,1);  % creates a string array

    J =( 0.1:0.01:maxJ)'; % vertical vector more resolution for next figs
    len = length(J);

    ang = repmat(A,length(J),1); % repeats the matrix A length(J) times in
                            % number rows and 1 time the number os columns

    dimension = size(ang); % returns length and width
    wide = dimension(2); % calls only width

    figCTsxJ = figure ('Name',strcat('CT x J - ',obj.Name));
        grid on
        hold on
      for k = 1:wide
        [CT] = FUN_CTax_CTwing_from_AoA_J(ang(:,k), J, fitCTxA_J_Surf ,...
                                                InterpSurf_wV_from_AoA_J);
        p = plot(J,CT,'Color',colormatrix(k,:),...
                            'HandleVisibility','on','LineWidth',1);
        dispname(k) = strcat('AoA= ',string(ang(1,k)),'\circ');
        set(p,'DisplayName',dispname(k))
      end

    set(gcf, 'units','centimeters','Position', [33 12 10.0 8.5]);
    set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

        set(gca,'XLim',[J(1) J(len)],'XTick',0.1:0.1:maxJ,...
            'XTickLabel',{'0.1','','0.3','','0.5','','0.7','','0.9',''},...
             'FontName', 'TimesNewRoman','FontSize',12,...
         'LabelFontSizeMultiplier',1,'xminorgrid','off','yminorgrid','off')

        legend('Location','southwest','FontName','TimesNewRoman',...
```

```
                                    'FontSize',8,'EdgeColor',[0.8 0.8 0.8])
        annotation('textbox',[0.40 0.71 0.3 0.2],'string',...
            obj.Name,'FitBoxToText','on','BackgroundColor',[1 1 1],...
               'FontName','TimesNewRoman','FontSize',9, ...
               'EdgeColor',[0. 0.45 0.74]);
        xlabel('J');
        ylabel('C_T');
        ylim([0 inf]);


%-----------------------------------------------------------------
        % 3RD FIGURE CT/J x J FOR ASSORTED AoAs  ok

    figCToverJxJ = figure ('Name',strcat(' CT_over_J x J - ',obj.Name));
        grid on
        hold on
      for k = 1:wide
        [CT] = FUN_CTax_CTwing_from_AoA_J(ang(:,k), J, fitCTxA_J_Surf ,...
                                                    InterpSurf_wV_from_AoA_J);
        p = plot(J,CT./J,'Color',colormatrix(k,:),...
                                    'HandleVisibility','on','LineWidth',1);
        dispname(k) = strcat('AoA= ',string(ang(1,k)),'\circ');
        set(p,'DisplayName',dispname(k))
      end
        xlabel('J')
        ylabel('C_T / J')
        ylim([0 inf]);

    set(gcf, 'units','centimeters','Position', [33 17 10.0 8.5]);
    set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

        set(gca,'XLim',[J(1) J(len)],'XTick',0.1:0.1:maxJ,...
        'XTickLabel',{'0.1','','0.3','','0.5','','0.7','','0.9',''},'FontName',...
                'TimesNewRoman','FontSize',12,'LabelFontSizeMultiplier',1,...
                'xminorgrid','off','yminorgrid','off')
        legend('Location','northeast','FontName','TimesNewRoman',...
                                    'FontSize',8,'EdgeColor',[0.8 0.8 0.8])
        annotation('textbox',[0.35 0.71 0.3 0.2],'string',...
            obj.Name,'FitBoxToText','on','BackgroundColor',[1 1 1],...
           'FontName','TimesNewRoman', 'FontSize',9, 'EdgeColor',...
           [0. 0.45 0.74]);

end


%-------------------------------------------------------------------------

function [figCTsxAoA, figsCT_CTa_CTw_xAoA] = Plot_CTs_x_AoA_variosJ_fromfit...
        (obj, fitCTxA_J_Surf ,InterpFunc_wV_from_AoA_J,optionalHorzVectorJs)

switch nargin   % allows for variable number of input arguments
    case 3      %  if last argument is not provided code choses may Js
        J = ( 0.1:0.15:1.2);   % horizontal vector
    case 4
        J = optionalHorzVectorJs; % MUST BE horizontal vector
    otherwise
        disp 'Wrong number of input arguments'
end

        black = [0 0 0];
        lgtblu  = [0 0.7 1];
        cyan=[0 1 1];
        magenta = [1 0 1];
        yellow = [0.95 1 0]; % a bit darker
        blue = [0 0 1];
        green = [0 1 0];
        red = [1 0 0];
```

```
        purp = [0.62  0.14  0.72];
        orng = [1  0.55  0.13];
        drkgren =[0  0.25  0];
        oilgrn = [0  0.5  0.5];
        lemon = [0.8  0.95  0];
        turquoise = [0  0.75  0.65];
        brwn = [0.5  0.35  0.15];

        colormatrix = [red;blue;lgtblu;orng;green;oilgrn;brwn;yellow;...
                                        cyan;purp;magenta;cyan;turquoise;black];
    A = (0:2:90)' ;
    len = length(A) ;

    advratio = repmat(J,length(A),1);% repeats the matrix J length(A) times
                          % in number rows and 1 time the number os columns

    dimension = size(advratio); % returns length and width
    wide = dimension(2); % recalls only width
    figsCT_CTa_CTw_xAoA = gobjects(1,wide); % creates an array of graphic
                                    % objects. must output horizontal vector
                    % graphic array for the script that analyses data to
                    % assemble many figures in a matrix to save.

  for k = 1:wide

    % ang(1,k);  % taking only the first element of the vector of AoA which
                    %  all values are the same. Length to match lengthof J
    [CT, CTax, CTwing] = FUN_CTax_CTwing_from_AoA_J(A, advratio(:,k), ...
                                    fitCTxA_J_Surf ,InterpFunc_wV_from_AoA_J);

    %-------------------------------------------------------------
    % 1ST SET OF FIGURES CT, CTaxial, CTwing x AoA FOR ASSORTED Js
%     figsCT_CTa_CTw_xAoA(k) = figure ('Name',...
%       strcat(obj.Name,' -  CT, CTax, CTw x AoA - J=',string(advratio(1,k))));

   % BELOW A COMMAND TO CHANGE DOT FOR COMMA SO FIG NAME WONT HAVE DOT IN IT
    % AND MATLAB WILL BE ABLE TO SAVE THE FIGURE IN EPS FORMAT
  Jstring = strrep(evalc('disp(advratio(1,k))'),'.',',');
      figsCT_CTa_CTw_xAoA(k) = figure ('Name',...
      strcat(obj.Name,' -  CT, CTax, CTw x AoA - J=',Jstring));
        grid on
        hold on

    plot(A,CT,'Color',oilgrn,'LineWidth',1)
    plot(A,CTax,'LineWidth',1)
    plot(A,CTwing,'Color',[0.53  0.78  0.95],'LineWidth',1)

    set(gcf, 'units','centimeters','Position', [40  17.5  10.0  8.5]);
    set(gca,'Units','centimeters','OuterPosition',[0.2  0  10.0  8.5]);
    set(gca,'FontName','TimesNewRoman','FontSize',12,'LabelFontSizeMultiplier',1)

    annotation('textbox',[0.48  0.71  0.3  0.2],'EdgeColor',[0.  0.45  0.74],...
                'string',strcat(obj.Name,'   J = ',string(advratio(1,k))),...
         'FontName','TimesNewRoman', 'FontSize',9,'FitBoxToText','on',...
         'BackgroundColor',[1  1  1])

    legend('C_T','C_T_{axial}','C_T_{wing}','EdgeColor',[0.8  0.8  0.8],...
           'FontName','TimesNewRoman','FontSize',8,'Location','Northwest')

    xlabel('AoA [deg]')
    ylabel('C_T')
    xlim([0  90]);
    xticks([0  15  30  45  60  75  90]);
    ylim([0  1.3*max(CT)]);
  end
%-------------------------------------------------------------
    % 2ND FIGURE CT x AoA FOR ASSORTED Js
```

```
    dispname = strings(wide,1);  % creates a string array

    figCTsxAoA = figure ('Name',strcat('CT x AoA - ',obj.Name));
        grid on
        hold on
      for k = 1:wide
        [CT] = FUN_CTax_CTwing_from_AoA_J(A, advratio(:,k),fitCTxA_J_Surf  ,...
                                                    InterpFunc_wV_from_AoA_J);
        p = plot(A,CT,'Color',colormatrix(k,:),...
                                'HandleVisibility','on','LineWidth',1);
        dispname(k) = strcat('J = ',string(advratio(1,k)));
        set(p,'DisplayName',dispname(k))  % for the legend
      end

    set(gcf, 'units','centimeters','Position', [40 10.5 10.0 8.5]);
    set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

        set(gca,'XLim',[0 90],'XMinorTick','on','YMinorTick','on',...
            'XMinorGrid','on','YMinorGrid','on','FontName',...
            'TimesNewRoman','FontSize',12,'LabelFontSizeMultiplier',1);
        legend('Location','northwest','FontName','TimesNewRoman',...
                                    'FontSize',8,'EdgeColor',[0.8 0.8 0.8])
        annotation('textbox',[0.46 0.71 0.3 0.2],'FitBoxToText','on',...
                   'string',obj.Name,'BackgroundColor',[1 1 1],...
                   'FontSize',9,'FontName','TimesNewRoman','EdgeColor',...
                   [0. 0.45 0.74]);
        xlabel('AoA [deg]')
        ylabel('C_T')
        ylim([0 inf]);
        xticks([0 15 30 45 60 75 90]);
end

%-------------------------------------------------------------------------

function [figsCT_withModelproj] = Plot_CTs_x_AoA_vs_SimplifiedMODEL...
                        (obj, fitCTxA_J_Surf ,InterpFunc_wV_from_AoA_J,...
                                        ModelAngleShift,optionalHorzVectorJs)
        % function that plots CT x AoA and CT projected model and CT
        % projected model with angle shift for comparison
switch nargin
    case 4
        J = ( 0.1:0.1:1.1);   % horizontal vector
    case 5
        J = optionalHorzVectorJs; % MUST BE horizontal vector
    otherwise
        disp 'Wrong number of input arguments'
end

        black = [0 0 0];
        lgtblu  = [0 0.7 1];
        cyan=[0 1 1];
        magenta = [1 0 1];
        burgundy = [0.64,0.08,0.18];

    A = (0:2:90)';
    len = length(A) ;

    advratio = repmat(J,length(A),1); % repeats the matrix A length(J)
                    % times in number rows and 1 time the number os columns

    dimension = size(advratio); % returns length and width
    wide = dimension(2); % returns only width
    figsCT_withModelproj = gobjects(1,wide); % creates an array of graphic
                                                            % objects
  for k = 1:wide

 %   ang(1,k);  % taking only the first element of the vector of AoA which
```

```
                    % all values are the same. Length to match length of J

    [CT] = FUN_CTax_CTwing_from_AoA_J(A, advratio(:,k),...
                                fitCTxA_J_Surf ,InterpFunc_wV_from_AoA_J);

    %-------------------------------------------------------------
    %  SET OF FIGURES CT x AoA, including CT projected model FOR
    %  ASSORTED Js

 %   figsCT_withModelproj(k) = figure ('Name',strcat(obj.Name,...
 %           ' - CT x AoA and CT projModel   J=',string(advratio(1,k))));
 % BELOW A COMMAND TO CHANGE DOT FOR COMMA SO FIG NAME WONT HAVE DOT IN IT
 % AND MATLAB WILL BE ABLE TO SAVE THE FIGURE IN EPS FORMAT
  Jstring = strrep(evalc('disp(advratio(1,k))'),'.',',');

   figsCT_withModelproj(k) = figure ('Name',strcat(obj.Name,...
            ' - CT x AoA and CT projModel   J=',Jstring));
        grid on
        hold on

    plot(A,CT,'Color',black,'LineWidth',1.2);

    CT_AoAo = fitCTxA_J_Surf (0,advratio(1,k));

    CT_projected_shift = propeller.CTmodel_analytic(CT_AoAo,...
                                            advratio(1,k),ModelAngleShift);
    fplot(CT_projected_shift,[0,90],'Color',[0.64,0.08,0.18],'LineWidth',1);

    CT_projected = propeller.CTmodel_analytic(CT_AoAo, advratio(1,k),0);
    fplot(CT_projected,[0,90],'Color',lgtblu,'LineWidth',1);

    set(gcf, 'units','centimeters','Position', [40 0.5 10.0 8.5]);
    set(gca,'Units','centimeters','OuterPosition',[0.2 0 10.0 8.5]);

    set(gca,'FontName','TimesNewRoman','FontSize',12,...
     'XTick',[0 15 30 45 60 75 90],'xlim',[0 90],'LabelFontSizeMultiplier',1)

    annotation('textbox',[0.5 0.71 0.3 0.2],'string',...
                        strcat(obj.Name,'  J=',string(advratio(1,k))),...
             'FontName','TimesNewRoman', 'FontSize',9,'FitBoxToText','on',...
             'EdgeColor',[0. 0.45 0.74],'BackgroundColor',[1 1 1]);

    str = strcat('C_T_{ Proj. Simple Model}',...
                            ' ang.shift=',string(ModelAngleShift),'\circ');
    legend('C_T',str,'C_T_{ Proj. Simple Model}','EdgeColor',[0.8 0.8 0.8],...
                                'FontName','TimesNewRoman','FontSize',8,...
                                                'Location','Northwest')
    xlabel('AoA [deg]')
    ylabel('C_T')
    xlim([0 90]);
    ylim([0.5*min(CT) 1.3*max(CT)]);
  end
%-------------------------------------------------------------

end

%-------------------------------------------------------------------------
  %  ERROR FIGURE. 1ST FIGURE IS A SET OF 8 SUBPLOTS AND 2ND FIGURE IS A
  %  SET OF 4SUBPLOTS

  function [Errorfigs] = ErrorPlots (obj,Table_ResultsAnalysed)

     FigureName1 = strcat(obj.Name,' - 8 errors analysis');
     Errorfigs(1)  = figure('Name',FigureName1);

     annotation('textbox',[0.365 0.87 0.3 0.1],'String',obj.Name,...
           'HorizontalAlignment','center','FontSize',10,'FitBoxToText',...
```

```
        'on','EdgeColor',[0. 0.45 0.74],'BackgroundColor',[1 1 1]);

set(gcf,'Units','centimeters','PaperType','<custom>','PaperPositionMode'...
    ,'manual','PaperPosition',[0 0 18 28],'PaperSize', [18 28] ,...
    'OuterPosition', [0 0 18 28])
 %-------------------------------------------------------------------------
     % PREPARING DATA FOR FITS AND PLOTS
    stdT        = Table_ResultsAnalysed.stdT;
    stdRPMpct   = Table_ResultsAnalysed.stdRPMpct;
    stdJpct     = Table_ResultsAnalysed.stdJpct;
    stdV        = Table_ResultsAnalysed.stdV;
    T           = Table_ResultsAnalysed.T;
    RPM         = Table_ResultsAnalysed.RPM;
    J           = Table_ResultsAnalysed.J;
    V           = Table_ResultsAnalysed.V;
    v=V(V>6); stdv=stdV(V>6); % ignoring V=0 data and noise around zero,
                              % reduces size of data points.
    jj          = J(stdJpct<30);
    stdjjpct    = stdJpct(stdJpct<30);% excluding outliers to plot, std>30%
    Rot         = RPM(stdRPMpct<7.6);
    stdRotpct   = stdRPMpct(stdRPMpct<7.6);
    Th          = T(stdT./T*100<50);
    stdTh       = stdT(stdT./T*100<50);
    Th          = Th(Th>0);
    stdTh       = stdTh(Th>0);
    cT          = Table_ResultsAnalysed.CT(Table_ResultsAnalysed.stdCTpct<15);
    cT          = cT(cT>=0);  % ignoring CT=0 data around zero
    stdcTpct    = Table_ResultsAnalysed.stdCTpct...
                                    (Table_ResultsAnalysed.stdCTpct<15);
    stdcTpct    = stdcTpct(cT>=0);
    ww          = Table_ResultsAnalysed.w...
                                    (Table_ResultsAnalysed.stdWpct<50);
    stdwpct     = Table_ResultsAnalysed.stdWpct...
                                    (Table_ResultsAnalysed.stdWpct<50);
  %-------------------------------------------------------------------------
  % 1st figure
  % subplot(m,n,p)  m=rows, n=columns, p number of subplot or axis
 m=4;   % number of rows
 n=2;   % number of columns

 ax1 = subplot(m,n,1);
    scatter(v,stdv./v*100,12,'LineWidth',0.8);
    xlabel('V [m/s]','FontSize',10);
    ylabel('\sigma_V / V  [%]','FontSize',10);
    annotation('textbox',[0.077 0.66 0.1 0.1],'String','a)','EdgeColor',...
               [1 1 1],'FontSize',10,'FitBoxToText','on','FontName',...
               'TimesnewRoman','FontWeight','bold','Color',[0 0.25 0])
    set(gca,'Box','off','XMinorGrid','on','YMinorGrid','on',...
        'FontName','TimesNewroman','FontSize',10,...
        'XMinorTick','off','YMinorTick','off') % could have been deleted
                              % as it is done at the end for all subplots
    grid on

 ax2=subplot(m,n,2);
    cdfplot(stdv./v*100);
    xlabel('\sigma_V / V  [%]');
    ylabel('CDF,  F(\sigma_V / V)');
    set(gca,'Box','off','XMinorGrid','on','YMinorGrid','on','YTick',...
        [0 0.25 0.5 0.75 1], 'YTickLabel',{'', '', '0.5', '' ,'1'})

    hold on
    title('')  % removes title that is default to cdfplot

    lognormErV = fitdist(stdv./v*100,'logNormal') ;
                                % fits data in stdT to a lognormal CDF
    x = linspace(0,max(ax1.YLim));
                        % defines maximum value of stdT to match the plot
```

```
    plot(x,cdf(lognormErV,x));

    str = char({'lognormal CDF fit ($\mu, s)$  ',...
                    strcat('$\mu\:=\:$',num2str(lognormErV.mu)), ...
                    strcat('$ s \:=\:$',num2str(lognormErV.sigma))});
    lgd = legend('empirical data CDF',str,'Location','southeast',...
                            'EdgeColor',[0.65,0.65,0.65],'FontSize',8);
    lgd.Interpreter = 'latex';
    annotation('textbox',[0.52 0.66 0.1 0.1],'String','b)','EdgeColor',...
                [1 1 1],'FontSize',10,'FitBoxToText','on','FontName',...
                'TimesnewRoman','FontWeight','bold','Color',[0 0.25 0])

ax3 = subplot(m,n,3);
    stdRotpct(isinf(stdRotpct))=[];% deletes Inf values of the vector data
    scatter(Rot,stdRotpct,12,'LineWidth',0.8 );
    xlabel('RPM');
    ylabel('\sigma_{RPM} / RPM  [%]');
    annotation('textbox',[0.077 0.443 0.1 0.1],'String','c)','EdgeColor',...
        [1 1 1],'FontSize',10,'FitBoxToText','on','FontName',...
         'TimesnewRoman','FontWeight','bold','Color',[0 0.25 0])
    grid on

ax4 = subplot(m,n,4);
    cdfplot(stdRotpct);
    xlabel('\sigma_{RPM} / RPM [%]');
    ylabel('CDF,  F(\sigma_{RPM} / RPM)');
    set(gca,'XMinorGrid','on','YMinorGrid','on','YTick',[0 0.25 0.5 0.75 1],...
        'YTickLabel',{'', '', '0.5', '' ,'1'})
    hold on
    title('')  % removes title that is default to cdfplot

    lognormErRPM = fitdist(stdRotpct,'logNormal') ;
                                % fits data in stdT to a lognormal CDF
    x = linspace(0,max(ax3.YLim));
                     % defines maximum value of stdT to match the plot
    plot(x,cdf(lognormErRPM,x));

    str = char({'lognormal CDF fit ($\mu, s)$  ',...
                strcat('$\mu\:=\:$',num2str(lognormErRPM.mu)),...
                strcat('$ s \:=\:$',num2str(lognormErRPM.sigma))});
    lgd = legend('empirical data CDF',str,'Location','southeast',...
       'EdgeColor',[0.65,0.65,0.65],'FontSize',8);
    lgd.Interpreter = 'latex';
    annotation('textbox',[0.52 0.443 0.1 0.1],'String','d)','EdgeColor',...
                [1 1 1],'FontSize',10,'FitBoxToText','on','FontName',...
                'TimesnewRoman','FontWeight','bold','Color',[0 0.25 0])

ax5 = subplot(m,n,5);
    scatter(jj,stdjjpct,12,'LineWidth',0.8);
    xlabel('J');
    ylabel('\sigma_J / J  [%]');
    set(gca,'XTick',[0.1 0.3 0.5 0.7 0.9 1.1])
    grid on;
    annotation('textbox',[0.077 0.22 0.1 0.1],'String','e)','EdgeColor',...
                [1 1 1],'FontSize',10,'FitBoxToText','on','FontName',...
                'TimesnewRoman','FontWeight','bold','Color',[0 0.25 0])

ax6 = subplot(m,n,6);
    yforfit = stdjjpct(~isinf(stdjjpct)); % ignores Inf values from J=0
    cdfplot(yforfit);
    xlabel('\sigma_J / J  [%]','FontSize',10);
    ylabel('CDF,  F(\sigma_J / J)','FontSize',10);
    set(gca,'XMinorGrid','on','YMinorGrid','on','YTick',[0 0.25 0.5 0.75 1],...
        'YTickLabel',{'', '', '0.5', '' ,'1'})
    hold on
    title('')  % removes title that is default to cdfplot
```

```
    lognormErJ = fitdist(yforfit,'logNormal') ;
                                % fits data in stdT to a lognormal CDF
    x = linspace(0,max(ax5.YLim));
                        % defines maximum value of stdT to match the plot
    plot(x,cdf(lognormErJ,x));

    str = char({'lognormal CDF fit ($\mu, s)$   ',strcat('$\mu\:=\:$',...
                num2str(lognormErJ.mu)),...
                strcat('$ s \:=\:$',num2str(lognormErJ.sigma))});
    lgd = legend('empirical data CDF',str,'Location','southeast',...
       'EdgeColor',[0.65,0.65,0.65],'FontSize',8);
    lgd.Interpreter='latex';
    annotation('textbox',[0.52 0.22 0.1 0.1],'String','f)','EdgeColor',...
              [1 1 1],'FontSize',10,'FitBoxToText','on','FontName',...
                'TimesnewRoman','FontWeight','bold','Color',[0 0.25 0])

ax7 = subplot(m,n,7);
    stdTh = stdTh(~isinf(stdTh./Th*100)); % ignores Inf values from T=0
    Th = Th(~isinf(stdTh./Th*100));
    scatter(Th,stdTh./Th*100,12,'LineWidth',0.8);
    xlabel('T [N]');
    ylabel('\sigma_T / T  [%]');
    grid on;
    annotation('textbox',[0.077 0.005 0.1 0.1],'String','g)',...
        'EdgeColor',[1 1 1],'FontSize',10,'FitBoxToText','on',...
        'FontName','TimesnewRoman','FontWeight','bold','Color',[0 0.25 0])

ax8 = subplot(m,n,8);
    cdfplot(stdTh./Th*100);
    xlabel('\sigma_T / T  [%]');
    ylabel('CDF,  F(\sigma_T / T)');
    set(gca,'XMinorGrid','on','YMinorGrid','on','YTick',[0 0.25 0.5 0.75 1],...
        'YTickLabel',{'', '', '0.5', '' ,'1'})
    hold on
    title('')  % removes title that is default to cdfplot

    lognormErT = fitdist(stdTh./Th*100,'logNormal') ;
                                % fits data in stdT to a lognormal CDF
    x = linspace(0,max(ax8.YLim));
                      % defines maximum value of stdT to match the plot
    plot(x,cdf(lognormErT,x));

    str = char({'lognormal CDF fit ($\mu, s)$   ',strcat('$\mu\:=\:$',...
        num2str(lognormErT.mu)),...
                strcat('$ s \:=\:$',num2str(lognormErT.sigma))});
    lgd = legend('empirical data CDF',str,'Location','southeast',...
       'EdgeColor',[0.65,0.65,0.65],'FontSize',8);
    lgd.Interpreter = 'latex';
    annotation('textbox',[0.52 0.005 0.1 0.1],'String','h)','EdgeColor',...
        [1 1 1],'FontSize',10,'FitBoxToText','on','FontName',...
         'TimesnewRoman','FontWeight','bold','Color',[0 0.25 0])

set(findobj(gcf,'type','axes'),'FontName','TimesNewRoman','Fontsize',10.5,...
    'FontWeight','Normal','LineWidth',0.5,'layer','bottom','Box','on',...
    'LabelFontSizeMultiplier',1,'XMinorGrid','on','YMinorGrid','on',...
    'XMinorTick','off','YMinorTick','off'); % instead of set(gca,.. for
                        % every ax, this will be valid for all subplots
                        % of the figure must be at the end
 %--------------------------------------------------------------------
      % 2nd figure
    FigureName2 = strcat(obj.Name,' - 4 errors analysis');
   Errorfigs(2) =  figure('Name',FigureName2);

      set(gcf,'Units','centimeters','PaperType','<custom>','PaperSize',...
            [18 14.5],'PaperPositionMode','manual','PaperPosition',...
                        [0 0 18 14.5],'OuterPosition',[0 0 18 14.5])
     % outerposition values tested to fit well in the page
```

```
    % page is A4/2 in height and narrower so it will fit well in Latex

      % subplot(m,n,p)  m=rows, n=columns, p number of subplot or axis
        m=2;   % number of rows
        n=2;   % number of columns

ax1 = subplot(m,n,1);
   scatter(cT,stdcTpct,12,'LineWidth',0.8);% already pre calculated
   xlabel('C_T','FontSize',10);
   ylabel('\sigma_C_T / C_T  [%]','FontSize',10);

   annotation('textbox',[0.077  0.46  0.1  0.1],'String','a)','EdgeColor',...
              [1  1  1],'FontSize',10,'FitBoxToText','on','FontName',...
            'TimesnewRoman','FontWeight','bold','Color',[0  0.25  0],...
                                                  'Units','normalized')
   set(gca,'ActivePositionProperty','position','Position',...
                   [0.133  0.5839  0.3282  0.3016],'Units','normalized')
   grid on

ax2 = subplot(m,n,2);
   cdfplot(stdcTpct);
   xlabel('\sigma_{C_T}/C_T [%]');
   ylabel('CDF,  F(\sigma_{C_T}/C_T)');
   hold on
   title('')  % removes title that is default to cdfplot

   lognormErCT = fitdist(stdcTpct,'logNormal') ;
                                % fits data in stdT to a lognormal CDF
   x = linspace(0,max(ax1.YLim));
                     % defines maximum value of stdT to match the plot
   plot(x,cdf(lognormErCT,x));

   str = char({'lognormal CDF fit ($\mu, s)$  ',strcat('$\mu\:=\:$',...
               num2str(lognormErCT.mu)),...
               strcat('$ s \:=\:$',num2str(lognormErCT.sigma))});
   lgd = legend('empirical data CDF',str,'Location','southeast',...
      'EdgeColor',[0.65,0.65,0.65],'FontSize',8);
   lgd.Interpreter = 'latex';
   annotation('textbox',[0.52  0.46  0.1  0.1],'String','b)','EdgeColor',...
       [1  1  1],'FontSize',10,'FitBoxToText','on','FontName',...
        'TimesnewRoman','FontWeight','bold','Color',[0  0.25  0],...
                                                  'Units','normalized')
   set(gca,'ActivePositionProperty','position','Position',...
            [0.5733  0.5839  0.3282  0.3016],'Units','normalized',...
            'XMinorGrid','on','YMinorGrid','on','YTick',...
            [0  0.25  0.5  0.75  1],'YTickLabel',{'', '', '0.5', '' ,'1'})

ax3 = subplot(m,n,3);
   scatter(ww,stdwpct,12,'LineWidth',0.8);
   xlabel('w [m/s]');
   ylabel('\sigma_w / w  [%]');
   annotation('textbox',[0.077  0  0.1  0.1],'String','c)','EdgeColor',...
              [1  1  1],'FontSize',10,'FitBoxToText','on','FontName',...
             'TimesnewRoman','FontWeight','bold','Color',[0  0.25  0],...
                                                  'Units','normalized')
   grid on
   set(gca,'ActivePositionProperty','position','Position',...
       [0.133  0.11  0.3282  0.3016],'Units','normalized')

ax4 = subplot(m,n,4);
   cdfplot(stdwpct);
   xlabel('\sigma_w / w  [%]');
   ylabel('CDF,  F(\sigma_w / w)');
   title('')  % removes title that is default to cdfplot
   hold on;

   lognormErW = fitdist(stdwpct,'logNormal') ;
```

```
                                    % fits data in stdT to a lognormal CDF
        x = linspace(0,max(ax3.YLim));
                                % defines maximum value of stdT to match the plot
        plot(x,cdf(lognormErW,x));

        str = char({'lognormal CDF fit ($\mu, s)$  ',strcat('$\mu\:=\:$',...
                    num2str(lognormErW.mu)),...
                    strcat('$ s \:=\:$',num2str(lognormErW.sigma))});
        lgd = legend('empirical data CDF',str,'Location','southeast',...
           'EdgeColor',[0.65,0.65,0.65],'FontSize',8);
        lgd.Interpreter = 'latex';
        annotation('textbox',[0.52 0 0.1 0.1],'String','d)','EdgeColor',...
                   [1 1 1],'FontSize',10,'FitBoxToText','on','FontName',...
                  'TimesnewRoman','FontWeight','bold','Color',[0 0.25 0],...
                  'Units','normalized')
        set(gca,'ActivePositionProperty','position','Position',...
                  [0.5733 0.11 0.3282 0.3016],'Units','normalized',...
                              'XMinorGrid','on','YMinorGrid','on','YTick',...
                  [0 0.25 0.5 0.75 1],'YTickLabel',{'', '', '0.5', '' ,'1'})

    set(findobj(gcf,'type','axes'),'FontName','TimesNewRoman','Fontsize',...
      10.5,'FontWeight','Normal','LineWidth',0.5,'layer','bottom','Box',...
      'on','LabelFontSizeMultiplier',1,'XMinorGrid','on','YMinorGrid','on');
          % instead of set(gca,.. for every ax, this will be valid for all
          % subplots of the figure must be at the end

    end
%-------------------------------------------------------------------------


%-------------------------------------------------------------------------
%         function w = get.w(obj)     % finds induced velocity ALTERNATIVELY
                                      % NOT VERY ROBUST TESTED
%
%             leng = length (obj.T);
%             w = zeros(leng,1);
%             for k = 1:1:leng
%                 if abs(obj.T(k)<=0.05)   % calculating only for  T>0
%                 w(k) = 0;
%                 else
%                     warning('off')
%               w(k) = propeller.find_w(obj.Din,obj.T(k),obj.V(k),obj.AoA(k));
%                 end
%             end
%             warning('on')
%         end

 %-------------------------------------------------------------------------



    end

end
```

Bibliography

[1] EnciclopaediaBritannica, *Technology/helicopter*. [Online]. Available: https://www.britannica.com/technology/helicopter (visited on 01/20/2021).

[2] W. J. Fredericks, M. D. Moore, and R. C. Busan, "Benefits of hybrid-electric propulsion to achieve 4x cruise efficiency for a vtol uav," in *2013 International Powered Lift Conference*, 2013, p. 4324.

[3] DefenseMediaNetwork, *The tail sitter xfy-1 pogo and xfv-1*. [Online]. Available: https://www.defensemedianetwork.com/stories/tail-sitter-xfy-1-pogo-xfv-1/ (visited on 01/20/2021).

[4] BAEsystems, *Hawker-siddeley-harrier*. [Online]. Available: baesystems.com/en/heritage/hawker-siddeley-harrier (visited on 01/20/2021).

[5] M. J. Hirschberg, *Soviet V/STOL Aircraft: The Struggle for a Shipborne Combat Capability*, 14. AIAA, 1997.

[6] Boeing, *V-22 osprey tiltrotor*. [Online]. Available: http://www.boeing.com/history/products/v-22-osprey.page (visited on 01/20/2021).

[7] LockheedMartin, *X2-defiant*. [Online]. Available: https://www.lockheedmartin.com/en-us/products/sb1-defiant-technology-demonstrator.html (visited on 01/20/2021).

[8] R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich, "Autonomous vehicle technologies for small fixed-wing uavs," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 1, pp. 92–108, 2005.

[9] R. G. Valenti, I. Dryanovski, and J. Xiao, "Keeping a good attitude: A quaternion-based orientation filter for imus and margs," *Sensors*, vol. 15, no. 8, pp. 19 302–19 330, 2015.

[10] M.-D. Hua, T. Hamel, P. Morin, and C. Samson, "Introduction to feedback control of underactuated vtolvehicles: A review of basic control design ideas and principles," *IEEE Control systems magazine*, vol. 33, no. 1, pp. 61–75, 2013.

[11] S. Verling, B. Weibel, M. Boosfeld, K. Alexis, M. Burri, and R. Siegwart, "Full attitude control of a vtol tailsitter uav," in *2016 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2016, pp. 3006–3012.

[12] N. B. F. da Silva and K. R. L. J. C. Branco, "A new concept of vtol as fixed-wing," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2013, pp. 811–817.

[13] R. Austin, *Unmanned aircraft systems: UAVS design, development and deployment*. John Wiley & Sons, 2011, vol. 54.

[14] P. Casau, D. Cabecinhas, and C. Silvestre, "Hybrid control strategy for the autonomous transition flight of a fixed-wing aircraft," *IEEE Transactions on control systems technology*, vol. 21, no. 6, pp. 2194–2211, 2012.

[15] DroneIndustriesInsight, *Drone market size - 2020-2025*. [Online]. Available: https://droneii.com/the-drone-market-size-2020-2025-5-key-takeaways (visited on 01/20/2021).

[16] FortuneBusinessInsight, *Aerospace & defense / military drone market*. [Online]. Available: https://www.fortunebusinessinsights.com/military-drone-market-102181 (visited on 01/20/2021).

[17] GrandViewResearch, *Market analysis report*. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/global-commercial-drones-market (visited on 01/20/2021).

[18] MeticulousResearch, *Unmanned aerial vehicle (uav) market*. [Online]. Available: https://www.meticulousresearch.com/product/unmanned-aerial-vehicle-UAV-market-5086 (visited on 01/20/2021).

[19] FinancialTimes, *How the commercial drone market became big business*. [Online]. Available: https://www.ft.com/content/cbd0d81a-0d40-11ea-bb52-34c8d9dc6d84 (visited on 01/20/2021).

[20] ElectricVTOLNews, *Evtol aircraft directory*. [Online]. Available: https://evtol.news/aircraft (visited on 01/20/2021).

[21] P. M. Rothhaar, P. C. Murphy, B. J. Bacon, I. M. Gregory, J. A. Grauer, R. C. Busan, and M. A. Croom, "NASA Langley Distributed Propulsion VTOL Tiltwing Aircraft Testing, Modeling, Simulation, Control, and Flight Test Development," in *14th AIAA aviation technology, integration, and operations conference*, 2014, p. 2999. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20140011413.pdf.

[22] DHL/AndreasHeddergott. [Online]. Available: https://www.dw.com/en/dhl-claims-its-drones-are-first-to-deliver/a-19245002 (visited on 01/20/2021).

[23] WingcopterGmbH. [Online]. Available: https://wingcopter.com/ (visited on 01/20/2021).

[24] WingtraAG. [Online]. Available: https://wingtra.com/ (visited on 01/20/2021).

[25] JobyAviation. [Online]. Available: https://www.jobyaviation.com/ (visited on 01/20/2021).

[26] H. C. McLemore and M. D. Cannon, "Aerodynamic Investigation of a Four-Blade Propeller Operating Through an Angle-of-Attack Range From 0° to 180°," NACA TN-3228, Langley Aeronautical Laboratory, Langley Field,Va, Tech. Rep., 1954.

[27] R. E. Kuhn and J. W. Draper, "Investigation of the Aerodynamic Characteristics of a Model Wing-propeller Combination and of the Wing and Propeller Separately at Angles of Attack up to 90°," NACA Report-1263, Langley Aeronautical Laboratory, Langley Field,Va, Tech. Rep., 1956. [Online]. Available: https://ntrs.nasa.gov/search.jsp?R=19930092264.

[28] P. F. Yaggy and V. L. Rogallo, "A wind-tunnel investigation of three propellers through an angle- of-attack range from 0° to 85°," Ames Research Center, NASA, Tech. Rep., 1960. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19890068078.pdf.

[29] H. S. Ribner, "Propellers in Yaw," NACA Report n.820 - Langley Memorial Aeronautical Laboratory, Langley Field, Va., Tech. Rep., 1943. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19930093307.pdf.

[30] J. De Young, "Propeller at high incidence," *J. Aircr.*, vol. 2, no. 3, pp. 241–250, 1965. DOI: https://doi.org/10.2514/3.43646.

[31] H. Glauert, ""Airplane Propellers"," in *Aerodyn. Theory - Vol IV, 1935*, W. F. Durand, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1935, ch. XII - Div. Pp. 169–359. DOI: https://doi.org/10.1007/978-3-642-91487-4_3. [Online]. Available: http://www.springerlink.com/index/10.1007/978-3-642-91487-4_3.

[32] H. Glauert, *On the horizontal flight of a helicopter*. London: HM Stationery Office, 1928.

[33] B. McCormick, *Aerodynamics, Aeronautics Flight Mechanics*. John Wiley Sons, Inc, 1979, ISBN: 0-471-03032-5.

[34] R. L. Rubin and D. Zhao, "New development of classical actuator disk model for propellers at incidence," *AIAA Journal*, vol. 59, no. 3, pp. 1040–1054, 2021. DOI: 10.2514/1.J059734.

[35] W. M. J. Rankine, "On the Mechanical Principles of the Action of Propellers," *Trans. Inst. Nav. Archit.*, vol. 6, pp. 13–39, 1865.

[36] R. E. Froude, "On the Part Played in Propulsion by Differences of Fluid Pressure," *Trans. Inst. Nav. Archit.*, vol. 30, pp. 390–405, 1889.

[37] J. Shapiro, *Principles of Helicopter Engineering*. New York: Mc Graw Hill Book Co., 1956, p. 16.

[38] D. O. Dommasch, *Elements of Propeller and Helicopter Aerodynamics*, L. Isaac Pitman Sons, Ed. London: Isaac Pitman Sons, Ltd, 1953.

[39] B. W. McCormick Jr, *Aerodynamics of V/STOL Flight*. New York, London: Academic Press Inc, 1967.

[40] A. Betz, "Eine Erweiterung der Schraubenstrahl Theorie," *Zeitschrift für Flugtechnik und Mot. 11*, vol. 11, pp. 105–110, 1920.

[41] S Drzewiecki, *Théorie générale de l'hélice: hélices aériennes et hélices marines*. Paris: Gauthier-Villars et cie, 1920, pp. 1–183.

[42] W. Johnson, *Helicopter theory*. Princeton, N.J, Princeton University Press.

[43] S. Goldstein, S. John, and K. Wilhelm, "On the Vortex Theory of Screw Propellers," *Math. Phys. Character*, vol. 123, no. 792, pp. 440–465, 1929. [Online]. Available: http://www.jstor.org/stable/95206http://about.jstor.org/terms.

[44] T. Theodorsen, *Theory of Propellers*. New York: McGraw-Hill Book Company, 1948.

[45] G. A. Van Kuik, J. N. Sørensen, and V. L. Okulov, "Rotor theories by Professor Joukowsky: Momentum theories," *Prog. Aerosp. Sci.*, vol. 73, pp. 1–18, 2015. DOI: https://doi.org/10.1016/j.paerosci.2014.10.001.

[46] V. L. Okulov, J. N. Sørensen, and D. H. Wood, "The rotor theories by Professor Joukowsky: Vortex theories," *Prog. Aerosp. Sci.*, vol. 73, pp. 19–46, 2015. DOI: https://doi.org/10.1016/j.paerosci.2014.10.002.

[47] H. S. Ribner, "Formulas for propellers in yaw and charts of the side-force Derivative," NACA Rep. n. 819 - Langley Memorial Aeronautical Laboratory, Langley Field, Va., Tech. Rep., 1943. DOI: 10.1021/ac60015a710. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19930093304.pdf.

[48] D. G. Ariza, "Study of the sensitivity to the lateral wind of a Mini Unmanned Aerial Vehicle with VTOL flight capabilities," Phd thesis, Institut Supérieur de l'Aéronautique et de l'Espace (ISAE), Toulouse, 2013.

[49] D. Serrano, M. Ren, A. J. Qureshi, and S. Ghaemi, "Effect of disk angle-of-attack on aerodynamic performance of small propellers," *Aerosp. Sci. Technol.*, vol. 92, pp. 901–914, 2019. DOI: https://doi.org/10.1016/j.ast.2019.07.022.

[50] W Haans, "Wind Turbine Aerodynamics in Yaw: Unravelling the measured rotor wake," Ph.D. dissertation, PhD thesis, TU Delft, The Netherlands, 2011, p. 271. [Online]. Available: http://resolver.tudelft.nl/uuid:57f0cea4-4e05-47bf-8f53-fb1d6e36d39f.

[51] M. Lin and F. Porté-Agel, "Large-Eddy Simulation of Yawed Wind-Turbine Wakes: Comparisons with Wind Tunnel Measurements and Analytical Wake Models," *Energies*, no. 23, p. 4574, DOI: https://doi.org/10.3390/en12234574.

[52] D. Micallef, T. Sant, A. G. Aissaoui, and A Tahour, "A review of wind turbine yaw aerodynamics"," in *Wind Turbines-Design, Control Appl.* IntechOpen, Ed., IntechOpen, 2016, ch. 2, ISBN: 978-953-51-2496-2. DOI: 10.5772/63445. [Online]. Available: http://dx.doi.org/10.5772/63445.

[53] S Øye, "Induced velocities for rotors in yaw," *Proc. Sixth IEA Symp. ECN, Petten*, 1992.

[54] H Rahimi, M Hartvelt, J Peinke, and J. G. Schepers, "Investigation of the current yaw engineering models for simulation of wind turbines in BEM and comparison with CFD and experiment," *J. Phys. Conf. Ser.*, p. 22 016, DOI: https://doi.org/10.1088/1742-6596/753/2/022016.

[55] H. Rahimi, A. M. Garcia, B. Stoevesandt, J. Peinke, and G. Schepers, "An engineering model for wind turbines under yawed conditions derived from high fidelity models," *Wind Energy*, no. 8, pp. 618–633, DOI: https://doi.org/10.1002/we.2182.

[56] J Schepers, "An engineering model for yawed conditions, developed on basis of wind tunnel measurements," *37th Aerosp. Sci. Meet. Exhib.*, p. 39, 1998. DOI: https://doi.org/10.2514/6.1999-39.

[57] J. G. Schepers, ""Engineering models in wind energy aerodynamics: Development, implementation and analysis using dedicated aerodynamic measurements"," Ph.D. dissertation, Phd thesis, TU Delft, The Netherlands, Delft, The Netherlands, 2012. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid:92123c07-cc12-4945-973f-103bd744ec87/datastream/OBJ/.

[58] A. Bramwell, ""Some Remarks on the Induced Velocity Field of a Lifting Rotor and on Glauert's Formula"," in *Aeronaut. Res. Counc. Curr. Pap. - C.P. No.1301*, London, 1974: H.M. Stationary Office, 1974, pp. 1–50.

[59] G. A. M. Van Kuik, "On the Revision of the Actuator Disc Momentum Theory," *Wind Eng.*, vol. 5, pp. 276–289, 1991. [Online]. Available: https://about.jstor.org/terms.

[60] G. A. M. Van Kuik, "The Edge Singularity of an Actuator Disc with a Constant Normal Load," *Proc. ASME Wind Energy Symp. Reno, Nevada,*, pp. 62–72, 2003. DOI: https://doi.org/10.1115/wind2003-356.

[61] P. M. Goorjian, "An invalid equation in the general momentum theory of actuator disk," *AIAA J.*, vol. 10, no. 4, pp. 543–544, 1972. DOI: https://doi.org/10.2514/3.50146.

[62] J Sørensen and R Mikkelsen, "On the Validity of the Blade Element Momentum Method," in *EWEC-CONFERENCE-*, P. Helm and A. Zervos, Eds., Copenhagen, 2001, pp. 362–366.

[63] G. A. M. Van Kuik and L. E. M. Lignarolo, "Potential flow solutions for energy extracting actuator disc flows," *Wind Energy*, vol. 19, no. 8, pp. 1391–1406, 2016. DOI: https://doi.org/10.1002/we.1902.

[64] R. Bontempo and M. Manna, "Verification of the axial momentum theory for propellers with a uniform load distribution," *Int. J. Turbomachinery, Propuls. Power*, vol. 4, no. 2, 2019. DOI: https://doi.org/10.3390/ijtpp4020008.

[65] J. N. Sørensen, *General momentum theory for horizontal axis wind turbines*, 4. Switzerland: Springer International Publishing, 2016, vol. 4, pp. 1–180, ISBN: 3-319-22113-2, 978-3-319-22113-7.

[66] R. Bontempo and M. Manna, "Analysis and evaluation of the momentum theory errors as applied to propellers," *AIAA J.*, vol. 54, no. 12, pp. 3840–3848, 2016. DOI: https://doi.org/10.2514/1.J055131.

[67] R. Bontempo and M. Manna, "Effects of the approximations embodied in the momentum theory as applied to the NREL PHASE VI wind turbine," *Int. J. Turbomachinery, Propuls. Power*, vol. 2, no. 2, 2017. DOI: https://doi.org/10.3390/ijtpp2020009.

[68] J. T. Conway, "Analytical solutions for the actuator disk with variable radial distribution of load," *J. Fluid Mech.*, vol. 297, pp. 327–355, 1995. DOI: https://doi.org/10.1017/S0022112095003120.

[69] J. T. Conway, "Exact actuator disk solutions for non-uniform heavy loading and slipstream contraction," *J. Fluid Mech.*, vol. 365, pp. 235–267, 1998. DOI: https://doi.org/10.1017/s0022112098001372.

[70] D. M. Pitt and D. A. Peters, "Theoretical Prediction of Dynamic-Inflow Derivatives," *Vertica*, vol. 5, no. No. 1, pp. 21–34, 1981. [Online]. Available: https://ci.nii.ac.jp/naid/80000912133/en/.

[71] J. A. Morillo and D. A. Peters, "Velocity field above a rotor disk by a new dynamic inflow model," *J. Aircr.*, vol. 39, no. 5, pp. 731–738, 2002. DOI: https://doi.org/10.2514/2.3019.

[72] A. Rosen and O. Gur, "Novel approach to axisymmetric actuator disk modeling," *AIAA J.*, vol. 46, no. 11, pp. 2914–2925, 2008. DOI: https://doi.org/10.2514/1.37383.

[73] S. Kominer and A. Rosen, "New actuator-disk model for a skewed flow," *AIAA J.*, vol. 51, no. 6, pp. 1382–1393, 2013. DOI: https://doi.org/10.2514/1.j052057.

[74] M. D. Patterson, "Conceptual Design of High-Lift Propeller Systems for Small Electric Aircraft," Phd Thesis, PhD thesis, Georgia Institute of Technology, GA, Atlanta, 2016.

[75] L. Veldhuis, "Propeller wing aerodynamic interference," PhD.Thesis, PhD thesis, TU Delft, Delft, The Netherlands, 2005, p. 418, ISBN: 9090195378. [Online]. Available: http://www.narcis.nl/publication/RecordID/oai:tudelft.nl:uuid:8ffbde9c-b483-40de-90e0-97095202fbe3.

[76] T. Van Holten, "Concentrator systems for wind energy, with emphasis on tipvanes," *Wind Eng.*, pp. 29–45, 1981.

[77] D. O. Dommasch, S. S. Sherby, and T. F. Connoly, *Airplane Aerodynamics*. New York: Pitman Aeronautical Publications, 1951, pp. 112 –122.

[78] R. L. Rubin and D. Zhao, "New Development of Classical Actuator Disk Model for Propellers at Incidence," *AIAA Journal*, pp. 1–15, 2020, ISSN: 0001-1452. DOI: 10.2514/1.J059734. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.J059734.

[79] D. Stevenson, "The university of canterbury wind tunnel," *New Zealand Engineering*, vol. 23, no. 10, pp. 403–407, 1968.

[80] JR3-Inc., *Jr3 load cell technologies*. [Online]. Available: https://jr3.com (visited on 01/20/2021).

[81] NationalInstruments, *LabVIEW version 16.0.0 2016*. Austin, Texas: National Instruments Corp., 2016.

[82] MonarchInstument., *Ros - remote optical led sensor*. [Online]. Available: https://monarchinstrument.com/collections/remote-sensors/products/remote-optical-led-sensor-with-8-ft-cable-and-mounting (visited on 01/20/2021).

[83] H Glauert, *The Elements of Airfoil Airscrew Theory*, 2nd. Cambridge: Cambridge University Press (CUP), 1948.

[84] MATLAB, *version 9.5.0.944444 (R2018b)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[85] H. W. Coleman and W. G. Steele, *Experimentation, validation, and uncertainty analysis for engineers*. John Wiley & Sons, 2018.

[86] HQprop. [Online]. Available: www.hqprop.com (visited on 01/20/2021).

[87] Team-BlackSheep. [Online]. Available: www.team-blacksheep.com (visited on 01/20/2021).

[88] T-MOTOR. [Online]. Available: https://uav-en.tmotor.com/ (visited on 01/20/2021).

[89] MATLAB, *Mathworks - lognormal distribution*. [Online]. Available: https://au.mathworks.com/help/stats/lognormal-distribution.html?searchHighlight=lognormal (visited on 01/20/2021).

[90] G. K. Ananda, M. S. Selig, and R. W. Deters, "Experiments of propeller-induced flow effects on a low-Reynolds-number wing," *AIAA J.*, vol. 56, no. 8, pp. 3279–3294, 2018. DOI: https://doi.org/10.2514/1.J056667.

[91] J. Brandt, R. Deters, G. Ananda, and M. Selig, *UIUC Propeller Database*. [Online]. Available: http://m-selig.ae.illinois.edu/props/propDB.html. (visited on 04/22/2020).

[92] R. W. Deters, "Performance and Slipstream Characteristics of Small Scale propellers at Low Reynolds Numbers," Phd Thesis, PhD thesis, University of Illinois at Urbana-Champaign, 2014, 2014.

[93] V. Hrishikeshavan and I. Chopra, "Performance, Flight Testing of a Shrouded Rotor Micro Air Vehicle in Edgewise Gusts," *J. Aircr.*, vol. 49, no. 1, pp. 193–205, 2012. DOI: https://doi.org/10.2514/1.C031477. [Online]. Available: http://arc.aiaa.org/doi/10.2514/1.C031477.

[94] T. C. Stokkermans, D. Usai, T. Sinnige, and L. L. Veldhuis, "Aerodynamic interaction effects between propellers in typical evtol vehicle configurations," *Journal of Aircraft*, pp. 1–19, 2021.

[95] T. C. Stokkermans and L. L. Veldhuis, "Propeller performance at large angle of attack applicable to compound helicopters," *AIAA Journal*, vol. 59, no. 6, pp. 2183–2199, 2021.

[96] APC-propellers. [Online]. Available: https://www.apcprop.com/ (visited on 01/20/2021).

[97] AEOrc. [Online]. Available: https://www.aeorc.com/ (visited on 01/20/2021).

[98] MasterAirscrewPropellers. [Online]. Available: https://www.masterairscrew.com/ (visited on 01/20/2021).

[99] CobraMotorsUSA. [Online]. Available: https://www.cobramotorsusa.com (visited on 01/20/2021).

[100] AXI-Model-Motors. [Online]. Available: https://www.modelmotors.cz/ (visited on 01/20/2021).